Программистам от программистов

Excel 2003 и VBA

Справочник программиста

Пол Киммел, Джон Грин, Стивен Буллен, Роб Боуви, Роберт Розенберг, Брайан Паттерсон

www.dialektika.com

Исправления, исходные коды и техническая поддержка на сайте
www.wrox.com

Programmer to Programmer™

Excel 2003 VBA

Programmer's Reference

Pa[illegible]n, Stephen Bullen, Rob Bovey, Robert Rosenberg, Brian D. Patterson

SPECIAL SALE 15% OFF SALE SPECIAL

Updates, source code, and Wrox technical support at **www.wrox.com**

Excel 2003 и VBA

Справочник программиста

Пол Киммел
Джон Грин
Стивен Буллен
Роб Боуви
Роберт Розенберг
Брайан Паттерсон

"Диалектика"
Москва • Санкт-Петербург • Киев
2006

Материал, защищенный авторским правом

ББК 32.973.26-018.2.75
К40
УДК 681.3.07

Компьютерное издательство "Диалектика"
Главный редактор *С.Н. Тригуб*
Зав. редакцией *В.Р. Гинзбург*
Перевод с английского и редакция *О.А. Лещинского*

По общим вопросам обращайтесь в издательство "Диалектика" по адресу:
info@dialektika.com, http://www.dialektika.com
115419, Москва, а/я 783; 03150, Киев, а/я 152

Киммел, Пол, **Грин**, Джон, **Буллен**, Стивен, **Боуви**, Роб, **Розенберг**, Роберт и др.

К40 Excel 2003 и VBA. Справочник программиста. : Пер. с англ. — М. : Издательский дом "Вильямс", 2006. — 1088 с. : ил. — Парал. тит. англ.

ISBN 5-8459-0921-X (рус.)

В данной книге рассматриваются вопросы проектирования и разработки приложений Excel с использованием встроенного языка VBA. Начав с основ VBA, вы познакомитесь с принципами автоматизации большинства выполняемых в Excel задач, а также с созданием надстроек, применением Windows API, профессиональными приемами отладки и обработки ошибок, использованием языка SQL для получения данных из внешних источников и программным управлением другими приложениями Office. В книге содержится множество примеров кода и, при необходимости, приводятся копии экранов. Все авторы являются признанными экспертами по разработке приложений для Excel, и их советы окажутся полезны как начинающим, так и опытным пользователям Excel, а также разработчикам.

ББК 32.973.26-018.2.75

Все названия программных продуктов являются зарегистрированными торговыми марками соответствующих фирм.

Никакая часть настоящего издания ни в каких целях не может быть воспроизведена в какой бы то ни было форме и какими бы то ни было средствами, будь то электронные или механические, включая фотокопирование и запись на магнитный носитель, если на это нет письменного разрешения издательства JOHN WILEY&Sons, Inc.

Copyright © 2006 by Dialektika Computer Publishing.

Original English language edition Copyright © 2004 by Wiley Publishing, Inc.

All rights reserved including the right of reproduction in whole or in part in any form. This translation is published by arrangement with Wiley Publishing, Inc.

ISBN 5-8459-0921-X (рус.) © Компьютерное изд-во "Диалектика", 2006, перевод, оформление, макетирование

ISBN 0-7645-5660-6 (англ.) © Wiley Publishing, Inc., 2004

Материал, защищенный авторским правом

Оглавление

Введение 31

Глава 1. Пример использования VBA в Excel 41

Глава 2. Программирование в редакторе VBE 103

Глава 3. Объект Application 115

Глава 4. Теория объектно-ориентированного программирования и VBA 129

Глава 5. Процедуры обработки событий 143

Глава 6. Модули классов 153

Глава 7. Создание надежного кода 167

Глава 8. Отладка и тестирование 191

Глава 9. Диалоговые окна UserForm 207

Глава 10. Добавление элементов управления 223

Глава 11. Доступ к данным с помощью ADO 237

Глава 12. Создание и использование надстроек 285

Глава 13. Надстройки Automation и надстройки COM 293

Глава 14. Настройка редактора VBE 321

Глава 15. Взаимодействие с другими приложениями Office 351

Глава 16. Программирование с помощью Windows API 365

Глава 17. Проблемы интернационализации 397

Глава 18. Книги и листы 433

Глава 19. Использование диапазонов 449

Глава 20. Использование имен 477

Глава 21. Работа со списками 489

Глава 22. Сводные таблицы 499

Глава 23. Списки с фильтрами 517

Глава 24. Генерация диаграмм 531

Глава 25. Файлы и папки Office 547

Глава 26. Командные панели 563

Глава 27. Смарт-теги 601

Глава 28. Excel и сеть Internet 625

Глава 29. XML и Excel 655

Приложение А. Объектная модель Excel 2003 665

Приложение Б. Объектная модель VBE 959

Приложение В. Объектная модель Office 2003 987

Предметный указатель 1074

Содержание

Введение 31

Глава 1. Пример использования VBA в Excel 41

Использование механизма записи макросов 42
Запись макроса 42
Личная книга макросов 45
Запуск макросов 46
Комбинации клавиш 46
Абсолютная и относительная запись 47
Редактор VBE 49
Модули кода 50
Процедуры 50
Окно проекта 51
Окно Свойства 51
Другие методы запуска макросов 52
Кнопки на листе 52
Панель инструментов Формы 52
Панель инструментов Control Toolbox 53
Панели инструментов 55
Процедуры обработки событий 57
Удаление присоединенной панели инструментов 58
Определенные пользователем функции 59
Создание определенных пользователем функций 59
Непосредственная ссылка на диапазоны 62
Чего не могут определенные пользователем функции 63
Объектная модель Excel 63
Объекты 63
Коллекции 64
Поля 65
Свойства 66
Методы 67
События 68
Получение справки 69
Окно Object Browser 70
Эксперименты в окне Immediate 71
Язык VBA 72
Базовый ввод и вывод 73
Передача параметров по позиции 74
Передача параметров по имени 74
Константы 76

Возвращаемые значения 76
Функция InputBox 77
Вызов функций и подпрограмм 78
Оператор Call 79
Объявление переменных 79
Оператор Option Explicit 79
Область видимости и время жизни переменных 81
Тип переменной 82
Определение типа переменной 84
Объявление функции и типы параметров 84
Константы 84
Соглашения по именованию переменных 85
Объектные переменные 85
Конструкция With... End With 86
Принятие решений 87
Оператор If 87
Блочный оператор If 88
Оператор Select Case 89
Циклы 90
Цикл While...Wend 91
Цикл Do...Loop 91
Цикл For... Next 93
Цикл For Each... Next 95
Массивы 95
Многомерные массивы 97
Динамические массивы 98
Обработка ошибок на этапе выполнения 99
Оператор On Error Resume Next 101
Резюме 102

Глава 2. Программирование в редакторе VBE 103

Написание кода 103
Программирование для людей 104
Написание кода в редакторе VBE 105
Куда делся мой код? 105
Управление проектом 106
Управление расположением элементов управления 107
Добавление классов 108
Модификация свойств 109
Импорт и экспорт кода Visual Basic 110
Редактирование 111
Управление параметрами редактора 111
Запуск и отладка кода 112
Использование контрольных значений 112

Использование окна Просмотр объектов 113
Резюме 114

Глава 3. Объект Application 115

Глобальные члены 115
Свойства Active 116
Вывод предупреждений 117
Обновление экрана 118
Оценка 118
Метод InputBox 120
Строка состояния 122
Свойство SendKeys 122
Метод OnTime 123
Метод OnKey 124
Функции листа 125
Свойство Caller 126
Резюме 127

Глава 4. Теория объектно-ориентированного программирования и VBA 129

Сравнение классов и интерфейсов 129
Определение интерфейса 131
Реализация интерфейса 131
Определение методов 132
Аргументы 133
Передача аргументов по значению 133
Передача аргументов по ссылке 134
Необязательные аргументы 134
Реализация рекурсивных методов 134
Отказ от рекурсии через использование циклов 135
Определение полей 135
Определение свойств 136
Свойства только для чтения 137
Свойства только для записи 137
Определение событий 137
Определение событий в классах 138
Создание события 139
Обработка событий 140
Сокрытие информации и квалификаторы доступа 140
Инкапсуляция, агрегация и ссылки 141
Резюме 142

Глава 5. Процедуры обработки событий 143
События листа 144
Включение событий 145
Событие Worksheet Calculate 145
События диаграммы 146
Событие BeforeDoubleClick 147
События книги 149
Сохранение изменений 150
Верхние и нижние колонтитулы 151
Резюме 152

Глава 6. Модули классов 153
Создание собственных объектов 154
Использование коллекций 155
Коллекция в модуле класса 156
Перехват событий приложения 158
Встроенные события диаграмм 160
Коллекция элементов управления UserForm 161
Ссылки на классы из других проектов 163
Резюме 165

Глава 7. Создание надежного кода 167
Использование метода Debug.Print 168
Использование метода Debug.Assert 169
Краткая история отладки на ПК 170
Создание многоразовых инструментов на основе объекта Debug 173
Определение последовательности выполнения 173
Получение маршрута выполнения кода 176
Проверка инвариантных условий 178
Вывод сообщений об ошибках 181
Создание обработчиков ошибок 183
Оператор On Error GoTo 183
Оператор On Error Resume Next 184
Оператор On Error GoTo 0 186
Использование объекта Err 186
Создание обвязки 186
Запись в журнал событий 188
Резюме 190

Глава 8. Отладка и тестирование 191
Пошаговое выполнение кода 192
Выполнение кода 192

Шаг с заходом 193
Шаг с обходом 194
Шаг с выходом 194
Выполнить до текущей позиции 194
Следующая инструкция 195
Отобразить следующую инструкцию 195
Использование точек останова 195
Использование контрольных значений 196
Добавление контрольного значения 196
Изменение контрольного значения 198
Контрольное значение 199
Окно Локальные переменные 199
Тестирование выражения в окне Проверка 200
Источники получения информации об определениях 200
Команда Краткие сведения 200
Команда Сведения о параметре 202
Команда Завершить слово 202
Команда Список свойств/методов 202
Команда Список констант 202
Команда Закладка 202
Команда Описания 203
Команда Просмотр объектов 203
Просмотр стека вызовов 204
Проверка инвариантных предположений 205
Резюме 205

Глава 9. Диалоговые окна UserForm 207

Отображение диалогового окна UserForm 207
Создание диалогового окна UserForm 209
Непосредственный доступ к элементам управления диалогового окна 211
Отключение кнопки Закрыть 215
Поддержка списка данных 215
Немодальные диалоговые окна UserForm 221
Резюме 221

Глава 10. Добавление элементов управления 223

Панели инструментов 223
Элементы управления ActiveX 224
Полоса прокрутки 225
Счетчик 226
Флажок 227
Переключатель 227
Элементы управления с панели Формы (Forms) 228

Динамические элементы управления ActiveX 232
Элементы управления, встроенные в диаграмму 235
Резюме 236

Глава 11. Доступ к данным с помощью ADO 237

Введение в структурированный язык запросов 238
Оператор SELECT 239
Оператор INSERT 240
Оператор UPDATE 241
Оператор CREATE TABLE 242
Оператор DROP TABLE 243
Обзор технологии ADO 243
Объект Connection 244
Свойства объекта Connection 245
Методы объекта Connection 247
События объекта Connection и асинхронное программирование 250
Коллекции объекта Connection 252
Объект Recordset 253
Свойства объекта Recordset 253
Методы объекта Recordset 254
События объекта Recordset 257
Коллекции объекта Recordset 257
Объект Command 258
Свойства объекта Command 259
Методы объекта Command 260
Коллекции объекта Command 262
Использование ADO в приложениях Microsoft Excel 262
Использование библиотеки ADO вместе с Microsoft Access 263
Подключение к Microsoft Access 263
Получение данных из базы данных Microsoft Access с помощью простого запроса 264
Получение данных из Microsoft Access с помощью хранимого запроса 266
Вставка, обновление и удаление записей в базе данных Microsoft Access с помощью простого текстового запроса SQL 267
Использование ADO вместе с Microsoft SQL Server 270
Подключение к Microsoft SQL Server 270
Хранимые процедуры Microsoft SQL Server 272
Несколько наборов записей 276
Отключенные наборы записей 277
Использование библиотеки ADO для доступа к нестандартным источникам данных 280
Запрос к книгам Microsoft Excel 280
Вставка и обновление записей в книгах Microsoft Excel 283
Запросы для текстовых файлов 283
Резюме 284

Глава 12. Создание и использование надстроек 285
Сокрытие кода 286
Преобразование книги в надстройку 287
Закрытие надстройки 289
Изменение кода 289
Сохранение изменений 290
Установка надстройки 290
Событие установки надстройки 291
Удаление надстройки из списка надстроек 292
Резюме 292

Глава 13. Надстройки Automation и надстройки COM 293
Надстройки Automation 293
Создание простой надстройки 294
Регистрация надстроек Automation в Excel 295
Регистрация через пользовательский интерфейс Excel 295
Создание ссылки на надстройку из кода VBA 295
Добавление надстройки посредством редактирования системного реестра 296
Использование надстроек Automation 298
Вызов функции из листа Excel 298
Вызов надстройки из кода VBA 299
Введение в интерфейс IDTExtensibility2 299
Надстройка Complex — генерация уникального случайного числа 302
Изменение порядка случайных чисел 305
Надстройки COM 308
Продолжение обзора интерфейса IDTExtensibility2 308
Регистрация надстройки COM в Excel 309
Конструктор надстроек COM 311
Подключение к Excel 312
Использование надстройки COM из кода VBA 316
Связывание с несколькими приложениями Office 317
Резюме 319

Глава 14. Настройка редактора VBE 321
Идентификация объектов редактора VBE 322
Объект VBE 322
Объект VBProject 322
Объект VBComponent 323
Объект CodeModule 324
Объект CodePane 324
Объект Designer 324
Начинаем 325
Добавление пунктов меню в редакторе VBE 326

Создание меню на основе таблиц 328
Вывод встроенных диалоговых окон, диалоговых окон UserForm и окон сообщений 335
Работа с кодом 340
Работа с диалоговыми окнами UserForm 344
Работа со ссылками 348
Резюме 350

Глава 15. Взаимодействие с другими приложениями Office 351

Установка подключения 352
Позднее связывание 352
Раннее связывание 354
Открытие документа в Word 356
Доступ к активному документу Word 357
Создание нового документа Word 357
Взаимодействие с Access через библиотеку DAO 358
Взаимодействие Access, Excel и Outlook 359
Когда вирус не является вирусом? 361
Резюме 362

Глава 16. Программирование с помощью Windows API 365

Анатомия вызова программного интерфейса приложений 366
Интерпретация объявлений в стиле C 367
Константы, структуры, обработчики и классы 370
Что делать, если что-то пошло не так? 373
Сокрытие вызовов API в модулях классов 374
Примеры классов 378
Класс таймера высокого разрешения 379
Модуль класса HighResTimer 379
Замораживание диалогового окна UserForm 380
Модуль класса FreezeForm 380
Класс информации о системе 382
Получение разрешения экрана (в пикселях) 382
Получение глубины цвета (в битах) 382
Получение ширины пикселя в координатах диалогового окна UserForm 383
Получение регистрационного идентификатора пользователя 383
Получение имени компьютера 383
Модификация стилей диалоговых окон UserForm 384
Свойства окон 384
Класс FormChanger 386
Диалоговые окна UserForm переменного размера 387
Абсолютные изменения 388
Относительные изменения 389

Класс FormResizer 389
Использование класса FormResizer 393
Другие примеры 394
Изменение пиктограммы Excel 394
Воспроизведение файла .wav 395
Резюме 395

Глава 17. Проблемы интернационализации 397

Изменение региональных параметров Windows и языка пользовательского интерфейса Office XP 398
Обработка региональных параметров и языка интерфейса Windows 398
Идентификация региональных параметров пользователя и языковой версии Windows 399
Функции преобразования VBA с точки зрения интернационализации 399
Неявное преобразование 399
Строки с датами 401
Функции IsNumeric и IsDate 401
Функция CStr 402
Функции CDbl, CSng, CLng, CInt, CByte, CCur и CDec 402
Функции CDate и DateValue 402
Функция CBool 402
Функция Format 402
Функции FormatCurrency, FormatDateTime, FormatNumber и FormatPercent 403
Функция Str 403
Функция Val 404
Функция Application.Evaluate 405
Взаимодействие с Excel 405
Отправка данных в Excel 406
Чтение данных из Excel 408
Правила работы с Excel 409
Взаимодействие с пользователями 410
Размер бумаги 410
Вывод данных 410
Интерпретация данных 411
Свойства xxxLocal 411
Правила работы с пользователями 412
Возможности интернационализации в Excel 2003 413
Возможности, не следующие общим правилам 415
Использование функции OpenText 416
Функция SaveAs 417
Подпрограмма ShowDataForm 418
Подпрограмма RunMenu 418
Вставка текста 419
Вычисляемые поля и элементы сводной таблицы, а также формулы условного форматирования 419

Web-запросы 420
Использование функции листа TEXT 420
Свойства Range.Value и Range.FormulaArray 421
Метод Range.AutoFilter 421
Метод Range.AdvancedFilter 422
Использование функций Application.Evaluate, Application.ConvertFormula и Application.ExecuteExcel4Macro 422
Обработка языковых параметров Office XP 422
Откуда извлекается текст? 423
Хранилище региональных параметров 423
Языковые параметры пользовательского интерфейса Office 423
Языковая версия Windows 424
Идентификация языковых параметров пользовательского интерфейса Office 424
Создание многоязыковых приложений 425
Рекомендуемый подход 425
Как хранятся строковые ресурсы 426
Работа в многоязыковой среде 427
Оставляйте свободное место 427
Использование объектов Excel 427
Использование функции SendKeys 428
Правила разработки многоязыковых приложений 429
Полезные функции 429
Реализация функции WinToNum 429
Реализация функции WinToDate 430
Реализация функции FormatDate 431
Реализация функции ReplaceHolders 431
Резюме 432

Глава 18. Книги и листы 433
Использование коллекции Workbooks 433
Создание книги 433
Сохранение активной книги 434
Активизация книги 434
Получение имени файла из полного пути 435
Файлы в том же каталоге 437
Перезапись существующей книги 437
Сохранение изменений 439
Коллекция Sheets 439
Листы 440
Методы Copy и Move 441
Группирование листов 443
Объект Window 444
Синхронизация листов 445
Резюме 447

Глава 19. Использование диапазонов 449
Методы Activate и Select 449
Свойство Range 451
Сокращенные ссылки на диапазоны 451
Диапазоны на неактивных листах 452
Свойство Range объекта Range 452
Свойство Cells 453
Использование свойства Cells в качестве параметра свойства Range 454
Диапазоны на неактивных листах 454
Дополнительная информация о свойстве Cells объекта Range 455
Ссылка на диапазон с единственным параметром 456
Свойство Offset 457
Свойство Resize 458
Метод SpecialCells 460
Поиск последней ячейки 460
Удаление чисел 462
Свойство CurrentRegion 462
Свойство End 464
Получение ссылки на диапазоны через свойство End 465
Суммирование диапазона 465
Свойства Columns и Rows 466
Области 467
Методы Union и Intersect 469
Пустые ячейки 469
Копирование значений между массивами и диапазонами 471
Удаление строк 473
Резюме 475

Глава 20. Использование имен 477
Именование диапазонов 478
Использование свойства Name объекта Range 479
Специальные имена 479
Хранение значений в именах 480
Хранение массивов 481
Сокрытие имен 482
Работа с именованными диапазонами 482
Поиск имени 483
Поиск имени диапазона 485
Определение имен, пересекающих диапазон 486
Резюме 488

Глава 21. Работа со списками 489
Создание списка 489
Сокращенные команды для списков 490
Сортировка и фильтрация списка 490
Создание диалогового окна UserForm на основе списка 491
Изменение размера списков 492
Перетаскивание маркера изменения размера в нижнем углу списка 492
Суммы столбцов 492
Преобразование списка в диапазон 493
Публикация списков 493
Публикация списка 495
Внесение изменений в список 495
Просмотр списка на сервере SharePoint 496
Удаление списка 497
Резюме 497

Глава 22. Сводные таблицы 499
Создание отчета для сводной таблицы 500
Коллекция PivotCaches 502
Коллекция PivotTables 503
Коллекция PivotFields 503
Коллекция CalculatedFields 506
Коллекция PivotItems 508
Группирование 508
Свойство Visible 511
Коллекция CalculatedItems 512
Сводные диаграммы 512
Внешние источники данных 514
Резюме 515

Глава 23. Списки с фильтрами 517
Структурирование данных 517
Команда Форма 518
Автофильтр 519
Пользовательский Автофильтр 520
Добавление раскрывающихся списков 520
Получение точной даты 523
Копирование видимых строк 524
Поиск видимых строк 525
Расширенный фильтр 527
Резюме 529

Глава 24. Генерация диаграмм 531
Листы диаграмм 532
Записанный макрос 532
Добавление листа диаграммы с помощью кода VBA 533
Встроенные диаграммы 534
Использование механизма записи макросов 535
Добавление встроенной диаграммы с помощью кода VBA 535
Редактирование рядов данных 536
Определение рядов диаграммы с помощью массивов 540
Преобразование диаграммы для использования массивов 542
Определение использованных в диаграмме диапазонов 543
Метки диаграмм 544
Резюме 546

Глава 25. Файлы и папки Office 547
Объект FileSearch 548
Свойство FoundFiles 550
Свойство PropertyTests 551
Коллекция FileTypes 552
Коллекция SearchScopes 553
Свойство ScopeFolder 554
Коллекция SearchFolders 555
Объект FileDialog 557
Коллекция FileDialogFilters 559
Коллекция FileDialogSelectedItems 559
Типы диалоговых окон 559
Метод Execute 559
Свойство MultiSelect 560
Резюме 561

Глава 26. Командные панели 563
Панели инструментов, панели меню и всплывающие меню 563
Встроенные командные панели Excel 566
Элементы управления любого уровня вложенности 569
Идентификаторы FaceId 572
Создание новых меню 574
Макрос OnAction 575
Передача значений параметров 576
Удаление меню 577
Создание панели инструментов 578
Контекстные меню 582
Отображение всплывающих командных панелей 585

Отключение командных панелей 587
Отключение комбинации клавиш для доступа к диалоговому окну Настройка 588
Создание командных панелей на основе таблиц 589
Резюме 599

Глава 27. Смарт-теги 601
Улучшения в механизме смарт-тегов 603
Библиотека типов Microsoft SmartTags 2.0 603
Смарт-тег FileName 604
Структура объекта SmartTag 605
Уникальный идентификатор смарт-тега 605
Класс Recognizer 607
Класс Actions 610
Реализация возможностей объектов SmartTag в Office 2003 616
Регистрация смарт-тега 618
Использование смарт-тега FileName 620
Управление смарт-тегами из кода VBA 621
Проверка активности класса распознавания 621
Удаление смарт-тега из диапазона 622
Добавление смарт-тега в диапазон 622
Проблемы, связанные с использованием смарт-тегов 623
Метод Recognize 623
Покрытие 624
Резюме 624

Глава 28. Excel и сеть Internet 625
Что за ажиотаж? 626
Использование сети Internet для хранения книг 626
Использование сети Internet в качестве источника данных 627
Открытие Web-страниц в качестве книг 627
Использование Web-запросов 628
Разбор Web-страницы для получения конкретной информации 631
Использование сети Internet для публикации результатов 633
Настройка Web-сервера 633
Сохранение листов в виде Web-страниц 634
Добавление интерактивности с помощью Web-компонентов 634
Использование сети Internet в качестве канала связи 636
Связь с Web-сервером 636
Передача данных от клиента к серверному приложению 637
Передача данных от серверного приложения к клиенту 637
Приложение Web-сервера — запись журнала ошибок 637
База данных Access для хранения журнала ошибок 638

Виртуальные каталоги 638
Страница ASP, записывающая данные в журнал ошибок 639
Страница ASP для просмотра журнала ошибок 641
XML 645
Словари XML 646
Схема XML-SS 647
Использование XSLT для преобразования XML 650
Резюме 653

Глава 29. XML и Excel 655
Что такое XML? 655
Что такое XSD? 656
Что такое XMLSS? 658
Импорт данных в формате XML 658
Black Jack: гибкость данных 659
Импорт файла XML 661
Что такое XSL и XSLT? 661
Экспорт листа в файл XML 663
Резюме 664

Приложение А. Объектная модель Excel 2003 665
Общие свойства коллекций и связанных с ними объектов 665
Общие свойства коллекций 665
Общие свойства объектов 666
Объекты Excel, их свойства, методы и события 666
Объект Addin и коллекция Addins 666
Общие свойства объектов Addin 666
Объект Adjustments 667
Объект AllowEditRange и коллекция AllowEditRanges 668
Объект Application 670
Коллекция Areas 691
Объект AutoCorrect 693
Объект AutoFilter 694
Объект AutoRecover 696
Объект Axis и коллекция Axes 697
Объект AxisTitle 700
Объект Border и коллекция Borders 702
Коллекция CalculatedFields 703
Коллекция CalculatedItems 704
Объект CalculatedMember и коллекция CalculatedMembers 704
Объект CalloutFormat 706
Объект CellFormat 707
Объект Characters 710
Объект Chart и коллекция Charts 711

Объект ChartArea 723
Объект ChartColorFormat 725
Объект ChartFillFormat 725
Объект ChartGroup и коллекция ChartGroups 727
Объект ChartObject и коллекция ChartObjects 730
Объект ChartTitle 735
Объект ColorFormat 736
Объект Comment и коллекция Comments 737
Объект ConnectorFormat 738
Объект ControlFormat 740
Объект Corners 742
Объект CubeField и коллекция CubeFields 743
Объект CustomProperty и коллекция CustomProperties 745
Объект CustomView и коллекция CustomViews 747
Объект DataLabel и коллекция DataLabels 748
Объект DataTable 752
Объект DefaultWebOptions 753
Объект Diagram 755
Объект DiagramNode и коллекция DiagramNodes 758
Объект DiagramNodeChildren 760
Объект Dialog и коллекция Dialogs 760
Объект DisplayUnitLabel 761
Объект DownBars 763
Объект DropLines 763
Объект Error и коллекция Errors 764
Объект ErrorBars 765
Коллекция ErrorCheckingOptions 766
Объект FillFormat 768
Объект Filter и коллекция Filters 770
Объект Floor 771
Объект Font 772
Объект FormatCondition и коллекция FormatConditions 773
Объект FreeformBuilder 775
Объект Graphic 776
Объект Gridlines 778
Коллекция GroupShapes 779
Объект HiLoLines 779
Объект HPageBreak и коллекция HPageBreaks 780
Объект Hyperlink и коллекция Hyperlinks 781
Объект Interior 783
Объект IRtdServer 784
Объект IRTDUpdateEvent 785
Объект LeaderLines 786
Объект Legend 786

Объект LegendEntry и коллекция LegendEntries 788
Объект LegendKey 789
Объект LineFormat 791
Объект LinkFormat 792
Объект ListColumn 793
Объект ListDataFormat 793
Объект ListObject 794
Объект ListRow 795
Объект Mailer 795
Объект Name и коллекция Names 796
Объект ODBCError и коллекция ODBCErrors 798
Объект OLEDBError и коллекция OLEDBErrors 799
Объект OLEFormat 800
Объект OLEObject и коллекция OLEObjects 801
Объект Outline 805
Объект PageSetup 806
Объект Pane и коллекция Panes 809
Объект Parameter и коллекция Parameters 810
Объект Phonetic и коллекция Phonetics 812
Объект PictureFormat 813
Объект PivotCache и коллекция PivotCaches 814
Объект PivotCell 817
Объект PivotField, коллекция PivotFields и коллекция CalculatedFields 818
Объект PivotFormula и коллекция PivotFormulas 823
Объект PivotItem, коллекция PivotItems и коллекция CalculatedItems 823
Объект PivotItemList 825
Объект PivotLayout 826
Объект PivotTable и коллекция PivotTables 827
Объект PlotArea 835
Объект Point и коллекция Points 837
Объект Protection 839
Объект PublishObject и коллекция PublishObjects 841
Объект QueryTable и коллекция QueryTables 843
Объект Range 847
Объект RecentFile и коллекция RecentFiles 860
Объект RoutingSlip 861
Объект RTD 862
Объект Scenario и коллекция Scenarios 863
Объект Series и коллекция SeriesCollection 864
Объект SeriesLines 869
Объект ShadowFormat 869
Объект Shape и коллекция Shapes 870
Объект ShapeNode и коллекция ShapeNodes 875
Коллекция ShapeRange 877

Коллекция Sheets 880
Объект SmartTag и коллекция SmartTags 882
Объект SmartTagAction и коллекция SmartTagActions 883
Коллекция SmartTagOptions 884
Объект SmartTagRecongnizer и коллекция SmartTagRecongnizers 884
Объект SoundNote 885
Объект Speech 885
Коллекция SpellingOptions 886
Объект Style и коллекция Styles 887
Объект Tab 889
Объект TextEffectFormat 890
Объект TextFrame 891
Объект ThreeDFormat 892
Объект TickLabels 894
Объект TreeviewControl 895
Объект Trendline и коллекция Trendlines 895
Объект UpBars 897
Коллекция UsedObjects 898
Коллекция UserAccess 898
Коллекция UserAccessList 899
Объект Validation 900
Объект VPageBreak и коллекция VPageBreaks 902
Объект Walls 903
Объект Watch и коллекция Watches 904
Объект WebOptions 905
Объект Window и коллекция Windows 906
Объект Workbook и коллекция Workbooks 911
Объект Worksheet и коллекция Worksheets 925
Объект WorksheetFunction 934
Объект XmlDataBinding 956
Объект XmlMap 957

Приложение Б. Объектная модель VBE 959

Связь между объектной моделью Excel и объектной моделью VBE 960
Общие свойства и методы 960
Объект AddIn и коллекция AddIns 961
Общие свойства объекта AddIn 961
Свойства объекта AddIn 962
Методы коллекции AddIns 962
Примеры использования надстроек 962
Объект CodeModule 963
Общие свойства объекта CodeModule 963
Свойства объекта CodeModule 964
Методы объекта CodeModule 965

Пример использования объекта CodeModule 966
Объект CodePane и коллекция CodePanes 967
Общие свойства объекта CodePane 967
Свойства объекта CodePane 967
Методы CodePane 968
Свойства коллекции CodePanes 968
Примеры использования объекта CodePane 968
Объект CommandBarEvents 969
События объекта CommandBarEvents 969
Примеры использования объекта CommandBarEvents 969
Объект Events 970
Свойства объекта Events 971
Примеры использования объекта Events 971
Коллекция LinkedWindows 971
Методы коллекции LinkedWindows 971
Объект Property и коллекция Properties 971
Общие свойства объекта Property 971
Свойства объекта Property 972
Примеры использования объекта Property 972
Объект Reference и коллекция References 973
Общие свойства объекта Reference 973
Свойства объекта Reference 973
Методы коллекции References 974
События коллекции References 974
Примеры использования объекта Reference 974
Объект ReferencesEvents 976
События объекта ReferencesEvents 977
Примеры использования объекта ReferencesEvents 977
Объект VBComponent и коллекция VBComponents 977
Общие свойства объекта VBComponent 978
Свойства объекта VBComponent 978
Методы объекта VBComponent 979
Методы коллекции VBComponent 979
Примеры использования объекта VBComponent 979
Объект VBE 980
Свойства объекта VBE 980
Примеры использования объекта VBE 981
Объект VBProject и коллекция VBProjects 981
Общие свойства объекта VBProject 981
Свойства объекта VBProject 982
Методы объекта VBProject 983
Методы коллекции VBProjects 983
Примеры использования объектов VBProject 983

Объект Window и коллекция Windows 984
Общие свойства объекта Window 984
Свойства объекта Window 984
Методы объекта Window 985
Методы коллекции Windows 985
Примеры использования объекта Window 985

Приложение В. Объектная модель Office 2003 987

Общие свойства коллекций и связанных с ними объектов 987
Общие свойства коллекций 987
Общие свойства объектов 988
Объекты пакета Office, их свойства и события 988
Объект AnswerWizard 988
Коллекция AnswerWizardFiles 989
Объект Assistant 990
Объект Balloon 994
Коллекция BalloonCheckBoxes 995
Объект BalloonCheckBox 996
Коллекция BalloonLabels 996
Объект BalloonLabel 996
Коллекция COMAddins 998
Объект COMAddin 999
Коллекция CommandBars 1000
Объект CommandBar 1002
Объект CommandBarButton 1005
Объект CommandBarComboBox 1009
Коллекция CommandBarControls 1014
Объект CommandBarControl 1015
Объект CommandBarPopup 1019
Объект DocumentLibraryVersion 1023
Коллекция DocumentLibraryVersions 1024
Коллекция DocumentProperties 1024
Объект DocumentProperty 1026
Объект FileDialog 1028
Коллекция FileDialogFilters 1031
Объект FileDialogFilter 1032
Коллекция FileDialogSelectedItems 1033
Объект FileSearch 1036
Коллекция FileTypes 1038
Объект FoundFiles 1039
Объект HTMLProject 1039
Коллекция HTMLProjectItems 1042
Объект HTMLProjectItem 1042
Объект LanguageSettings 1043

Объект MsoEnvelope 1044
Объект NewFile 1046
Коллекция ODSOColumns 1047
Объект ODSOColumn 1047
Коллекция ODSOFilters 1048
Объект ODSOFilter 1048
Объект OfficeDataSourceObject 1049
Объект Permission 1049
Коллекция PropertyTests 1050
Объект PropertyTest 1052
Коллекция ScopeFolders 1053
Объект ScopeFolder 1054
Коллекция Scripts 1057
Объект Script 1058
Коллекция SearchFolders 1059
Коллекция SearchScopes 1060
Объект SearchScope 1061
Объект SharedWorkspace 1062
Объект SharedWorkspaceFile 1063
Объект SharedWorkspaceFolder 1063
Объект SharedWorkspaceLink 1064
Объект SharedWorkspaceMember 1065
Коллекция SharedWorkspaceMembers 1065
Объект SharedWorkspaceTask 1065
Коллекция SharedWorkspaceTasks 1066
Объект Signature 1066
Коллекция SignatureSet 1067
Объект SmartDocument 1069
Объект Sync 1069
Объект UserPermission 1070
Коллекция WebPageFonts 1070
Объект WebPageFont 1071

Предметный указатель 1074

Об авторах

Пол Киммел

В 1990 году Пол Киммел (Paul Kimmel) основал компанию Software Conceptions, Inc. и с тех пор занимался проектированием и созданием программного обеспечения, а также писал книги на компьютерную тематику. Он является автором нескольких книг, посвященных VBA, VB, VB.NET, C#, Delphi и C++, кроме этого, раз в два месяца пишет статьи в колонку VB Today на сайте `www.codeguru.com` и часто публикуется в периодических и сетевых изданиях, включая `www.InformiT.com`. С Полом Киммелом можно связаться по электронной почте по адресу `pkimmel@softconcepts.com`.

Стивен Буллен

Стивен Буллен (Stephen Bullen) живет в Карлоу, Ирландия, и в Лондоне, Англия. Работает в собственной компании Business Modelling Solutions, Ltd. с 1997 года, специализируясь на разработках и консультациях по Excel. Стивен имеет опыт сотрудничества с крупнейшими мировыми компаниями. На сайте компании BMS по адресу `www.BMSLtd.co.uk` доступно большое количество его разработок, включая инструменты и утилиты для расширения функциональности Excel и множество примеров разработки приложений для Excel.

Значительную часть своего свободного времени Стивен уделяет поддержке других пользователей Excel, отвечая на вопросы на форуме CompuServe, посвященном Excel, и в Internet-конференциях компании Microsoft. Признавая его вклад и учитывая уровень знаний, начиная с 1996 года компания Microsoft ежегодно награждает Стивена званием Most Valuable Professional.

Стивен написал большую часть последних глав в книге *Excel 2000 VBA Programmer's Reference* и *Excel 2002 VBA Programmer's Reference*. Переработанные и обновленные Полом Киммелом фрагменты этих книг были интегрированы в настоящее издание. Стивен не принимал непосредственного участия в работе над этой книгой.

Джон Грин

Джон Грин (John Green) живет в Сиднее, Австралия, и работает независимым консультантом, специализируясь на Excel и Access. Имея 30-летний опыт работы с компьютерами, диплом инженера по химии и степень бакалавра, он применяет свои разносторонние знания в работе. Джон ведет тренинги по приложениям и операционным систе-

мам как в Австралии, так и за ее пределами. Начиная с 1995 года компания Microsoft ежегодно награждает Джона званием Most Valuable Professional.

Джон был основным автором книг *Excel 2000 VBA Programmer's Reference* и *Excel 2002 VBA Programmer's Reference.* Его материал был использован Полом Киммелом при создании этой книги. Джон не принимал непосредственного участия в работе над ней.

Роб Боуви

Роб Боуви (Rob Bovey) разрабатывает программное обеспечение и специализируется на приложениях для Microsoft Office, Visual Basic и SQL Server. Он является основателем и президентом компании Application Professionals, занимающейся разработкой приложений. Роб разработал несколько надстроек, которые распространяются компанией Microsoft в составе пакета Excel. Кроме этого, Роб являлся соавтором пакета Microsoft Excel 97 Developer's Kit. Начиная с 1995 года компания Microsoft ежегодно награждает Роба званием Most Valuable Professional. Роб является автором главы о доступе к данным через ADO в книге *Excel 2002 VBA Programmer's Reference*, но непосредственного участия в создании этой книги он не принимал.

Роберт Розенберг

Роберт Розенберг (Robert Rosenberg) работает в собственной консалтинговой компании, специализирующейся на предоставлении решений и обучении работе с продуктами Microsoft Office. Среди его клиентов можно заметить компании, входящие в список Fortune 500. Как обладатель звания Most Valuable Professional в области Excel он постоянно оказывает расширенную интерактивную поддержку пользователям Excel от лица компании Microsoft в ее Internet-конференциях. Роберт отвечал за обновление предметных указателей по Excel и Office для редакции книги 2002-го года. Это также касается обновления кода примеров и листингов для существующих объектов VBA, листингов описания новых объектов, методов, свойств и аргументов, а также примеров кода.

Брайан Паттерсон *(соавтор)*

На данный момент Брайан Паттерсон (Brian Patterson) работает в компании Illinois Mutual Life в качестве координатора разработки программного обеспечения, уделяя основное внимание использованию C# в WinForms и корпоративному Internet-сайту. Брайан написал множество статей с 1994 года. Он также является соавтором нескольких книг, посвященных технологии .NET, включая "Migrating to Visual Basic.NET" и ".NET Enterprise Development with VB.NET". Обычно его можно найти в MSDN Newsgroups или вместе с женой и тремя детьми. С Брайаном можно связаться по электронной почте по адресу `bdpatterson@illinoismutual.com`.

Благодарности

Пол Киммел

Я хотел бы поблагодарить хорошего друга и редактора Шерон Кокс (Sharon Cox), а также Кэти Мор (Katie Mohr) и Адаоби Оби Тултон (Adaobi Obi Tulton) из издательства Wiley. Без них и Дэвида Фьюгейта (David Fugate) (моего агента из компании Waterside), а также всех авторов книги я не смог бы работать над этим проектом. Работа с профессионалами из издательства Wiley доставила мне настоящее удовольствие.

Работая над этим проектом, я одновременно принял участие в проекте по C# в компании Pitney Bowes. Особая благодарность Эдварду Ронга (Edward Ronga), проявившему себя в качестве отличного менеджера; он совершал чудеса вместе с инженерами, которые иногда вели себя неадекватно. В компании Pitney мне довелось работать с такими людьми, как Леонард Бертелли (Leonard Bertelli), Джей Фуско (Jay Fusco), Энцо Маини (Enzo Maini), Питер Гомис (Peter Gomis), Кип Стробл (Kip Stroble), Карл Далзелл (Carl Dalzell), Дебра Алберти (Debra Alberti), Санжей Гулати (Sanjay Gulati) и мой сосед по комнате Чарльз Хейли (я слушал Нору Джонс четыре месяца, и Чарльз ни разу не пожаловался). Я стараюсь все оставлять в лучшем состоянии, чем нашел, и мне всегда в этом помогают новые хорошие взаимоотношения.

Хочу поздравить с Новым годом всех новых и старых друзей. Эрик Коттер (Eric Cotter) обладает более острым умом и большим энтузиазмом, чем все, кого я знаю, а Роберт Голиб (Robert Golieb) всегда готов к действию и является тем человеком, которого мне приятно называть другом. Хочу передать привет знакомым в заведении Порки в Шелтоне, штат Коннектикут, воспоминания о которых всегда ассоциируются с теплым приемом, отличными куриными крылышками и алкогольными напитками.

Самую большую благодарность я хочу выразить своей семье. Моя жена Лори всегда была для меня опорой и вдохновением. Еще мне повезло, что я имею четырех здоровых и красивых детей (Тревора, Дугласа, Алекса и Ноа). Больше всего я желаю, чтобы все мои знакомые имели благословение в виде любимой здоровой семьи.

Стивен Буллен

Я хотел бы начать раздачу благодарностей с покупателей книги *Excel 2002 VBA Programmer's Reference*, а также читателей, отправивших предложения и поздравления с выходом этого издания по электронной почте. Именно ваша поддержка позволила провести обновление книги до версии Excel 2002. Кроме этого, я хотел бы поблагодарить Джона Грина (John Green) за то, что он согласился выступить соавтором нового издания, Роба Боуви (Rob Bovey) за главу, посвященную ADO, и Роберта Розенберга (Robert Rosenberg) за работу над разделом справочника. Как всегда, сотрудники издательства Wrox Press и технические редакторы просто совершили чудо. Именно их вклад сделал возможным создание этой отличной книги.

Со своей стороны я хотел бы посвятить главы специалистам-офтальмологам, докторам и медсестрам в Temple Street Children's Hospital в Дублине за то, что они помогли справиться Джейн с опухолью, а также всем членам команды Ford ProPrima за их дружбу и поддержку в течение последнего года.

Роб Боуви

Я хотел бы поблагодарить свою жену Мишель за то, что она мирилась с моими компьютерными привычками, и свою собаку Харли за согреваемые во время работы ноги.

Роберт Розенберг

Я хотел бы поблагодарить Джона Грина, Стивена Буллена и Роба Боуви за то, что меня пригласили в один проект с тремя самыми значительными специалистами по Excel на данный момент. Этот вызов был для меня честью.

Особую благодарность хотелось бы выразить Робу Боуви, моему учителю, который всегда помогал мне, отвечал на все вопросы, учил множеству приемов и предоставил возможность поучаствовать в создании этой книги; а также моему лучшему другу и брату Эллиоту, который прошел со мной огонь и воду; и наконец, маме и папе за бесконечную любовь.

Введение

Впервые программа Excel появилась на платформе Macintosh в 1985 году и никогда не теряла место самого популярного приложения электронных таблиц в среде Macintosh. В 1987 году она была перенесена на ПК для работы под управлением операционной системы Windows. Вытеснение с рынка Lotus 1-2-3 (одна из наиболее успешных программных систем в истории персональных компьютеров на тот момент) заняло много лет.

До появления платформы IBM PC существовало несколько популярных приложений электронных таблиц. Это были VisiCalc, Quattro Pro и Multiplan. Все началось с VisiCalc, но этот продукт оказался на обочине достаточно рано. Multiplan выпускался компанией Microsoft до выхода Excel. В этом продукте для адресации ячеек использовался формат R1C1, который до сих пор доступен в Excel. Но именно Lotus 1-2-3 вырвался на вершину славы сразу после выхода в 1982 году и стал доминировать на рынке приложений электронных таблиц для ПК.

Ранние формы макросов для электронных таблиц

Пакет 1-2-3 был первым приложением электронных таблиц, в котором предоставлялась возможность использования электронных таблиц, диаграмм и механизмов баз данных в пределах одного пакета. Но основной причиной успеха являлась возможность создания и использования макросов. Существует легенда, что разработчики создали макросы в качестве инструмента отладки и тестирования продукта. Считается, что потенциал макросов был оценен в последний момент и возможность их использования была включена в состав продукта в конце цикла разработки.

Несмотря на происхождение, макросы предоставляют непрограммистам простой способ стать программистами и автоматизировать собственные электронные таблицы. Пользователи ухватились за эту возможность и "побежали". Наконец они получили определенную степень свободы от компьютерного отдела.

Оригинальные макросы 1-2-3 выполняли поставленную задачу, повторяя последовательность нажатий клавиш, которая позволяла выполнить данную задачу вручную. Таким образом, для создания макроса не нужно было учиться ничему новому и можно было сразу переходить от ручного управления к программной манипуляции таблицами. Достаточно было запомнить и записать последовательность нажатий клавиш. Единственным движением в сторону традиционного программирования были восемь дополнительных команд `/x`. Команды `/x` предоставляли примитивный механизм принятия решений и ветвления, получения данных от пользователя и способ создания меню.

Одной из основных проблем при применении макросов 1-2-3 являлась их уязвимость. Книга из нескольких листов еще не была изобретена, и макросы должны были записываться непосредственно в ячейки поддерживаемой электронной таблицы, рядом с данными и расчетами. Макросы отдавались на милость пользователя, который мог случайно повредить их, вставив или удалив строки или столбцы. Кроме этого, макросы оказывались под полным контролем программиста. Некорректно спроектированный макрос мог самоуничтожиться в результате попытки отредактировать данные в электронной таблице.

Несмотря на проблемы, пользователи наслаждались возможностями программирования и на этом сложном языке были написаны миллионы строк кода, в которых применялись загадочные методики, позволяющие обойти ограничения языка. Весь мир зависел от плохо продуманного кода, который практически всегда был плохо документирован и очень уязвим, хотя использовался для поддержки критических систем управления.

Язык макросов XLM

Для использования оригинального языка макросов Excel они записывались на специальном листе, сохраняемом в файле с расширением `.xlm`. Таким образом, макросы хранились отдельно от листа электронной таблицы, который сохранялся в файле с расширением `.xls`. Часто для обозначения этих макросов использовалось название XLM или макросы Excel 4, что позволяло отличить их от макросов на языке VBA, предоставленном в составе Excel 5.

Язык макросов XLM состоял из вызовов функций, расположенных в столбцах на листе макросов. Сотни функций обеспечивали доступ ко всем возможностям Excel и допускали программное управление. Язык XLM был более сложным и мощным, чем макроязык 1-2-3, даже с учетом улучшений, которые были внесены во 2 и 3 версии. Но код на этом языке был таким же сложным и запутанным.

Сложность макроязыка Excel была обоюдоострым мечом. Он хорошо подходил специалистам с хорошими навыками программирования, но был недоступен для понимания большинству пользователей. Не существовало простой связи между ручным управлением электронными таблицами Excel и методами программирования. Язык XLM имел очень крутую кривую обучения и высокий порог входа.

Еще одним препятствием для широкого распространения Excel на ПК являлась необходимость использования операционной системы Windows. Ранние версии Windows страдали от ограниченного доступа к оперативной памяти и больших требований к аппаратным средствам по сравнению с операционной системой DOS. Графический интерфейс пользователя выглядел привлекательно, но накладные расходы в виде стоимости аппаратных средств и снижения производительности рассматривались как серьезное препятствие..

Компания Lotus совершила ошибку, предположив, что операционная система Windows не удержится на рынке достаточно долго и будет замещена операционной системой OS/2, поэтому портирование 1-2-3 на операционную систему Windows не планировалось и основное внимание уделялось версии 1-2-3/G с графическим интерфейсом, которая работала под управлением операционной системы OS/2. Ставка на единственную лошадь оказалась причиной неудачи продукта 1-2-3 в борьбе за рынок.

К тому моменту, когда стало ясно, что операционная система Windows продолжит свое существование, компания Lotus оказалась в неприятной ситуации, наблюдая массовый переход пользователей на Excel. Первая попытка создания 1-2-3 для операционной системы Windows в 1991 году на самом деле являлась версией 3 для DOS в графической

оболочке. Последующие версии позволили сократить разрыв между 1-2-3 и Excel, но этот рывок начался слишком поздно, чтобы остановить практически повсеместный переход на пакет Microsoft Office.

Excel 5

Компания Microsoft приняла смелое решение по унификации программного кода приложений Office, представив язык VBA (Visual Basic for Applications) в виде общего макроязыка для приложений в составе пакета Office. Выпущенная в 1993 году Excel 5 была первой программой, в составе которой предоставлялась поддержка макроязыка VBA. Постепенно поддержка этого языка появилась и в других приложениях Microsoft Office. В составе Office XP язык VBA используется в таких приложениях, как Excel, Word, Access, PowerPoint, FrontPage, Visio, Project и Outlook. (Своими действиями компания Microsoft демонстрирует стремление расширять поддержку языка VBA в предлагаемых продуктах.)

С момента выхода Excel 5 одновременная поддержка языков XLM и VBA сохраняется и будет сохраняться в обозримом будущем, но значение этой поддержки будет снижаться наряду с переходом потребителей на использование языка VBA.

VBA является объектно-ориентированным языком программирования. По структуре и способам работы с объектами этот язык идентичен языку Visual Basic 6.0. В будущих версиях языка VBA можно будет наблюдать сближение этого языка с языком Visual Basic .NET. Изучив язык VBA в Excel, можно использовать этот язык и в других приложениях Office.

Разные приложения Office предоставляют различные объекты для использования в VBA. Для программирования приложения необходимо познакомиться с *объектной моделью (object model)*, представляющей собой иерархию всех объектов, которые доступны в пределах приложения. Например, фрагмент объектной модели Excel описывает объект `Application`, в составе которого доступен объект `Workbook`, содержащий объект `Worksheet`. В составе объекта `Worksheet` доступен объект `Range`.

Язык VBA изучается немного проще, чем макроязык XLM. Кроме этого, он предоставляет больше возможностей, обычно более эффективен и позволяет создавать хорошо структурированный код. Язык позволяет писать и плохо структурированный код, но придерживаясь нескольких простых принципов, можно по крайней мере создавать понятный другим разработчикам простой в сопровождении код.

В Excel 5 код VBA записывался в модули, которые являлись листами книги. В составе книг Excel 5 можно было создавать листы, листы диаграмм и диалоговые листы.

> На самом деле модуль является документом текстового процессора со специальными символами форматирования, позволяющими писать и тестировать код.

Excel 97

Вместе с Excel 97 компания Microsoft предоставила несколько значительных изменений в интерфейсе VBA и в объектной модели Excel. Начиная с Excel 97 модули не видны в окне приложения Excel и больше не являются объектами в составе объекта `Workbook`. Модули содержатся в проекте VBA, который связан с книгой и может просматриваться и редактироваться только в окне Visual Basic Editor (VBE).

Кроме стандартных модулей, были введены модули классов, позволяющие создавать собственные классы. Вместо меню и панелей инструментов были введены командные панели (объекты CommandBar), а диалоговые окна UserForm (объекты UserForm) заменили диалоговые листы. Как и модули, диалоговые окна UserForm могут редактироваться только в окне VBE. Как обычно, замененные объекты все еще поддерживаются в Excel, но рассматриваются как скрытые. В справочном руководстве отсутствует информация о скрытых объектах.

В предыдущих версиях Excel такие объекты, как встроенные в лист кнопки, реагировали только на одно событие. Обычно это было событие Click. В Excel 97 значительно расширено количество событий, на которые может отвечать код VBA. Кроме этого, была формализована процедура обработки событий. Для этого были предоставлены обработчики событий для объектов книги, листа и листа диаграммы. Теперь книга может реагировать на 20 событий, например BeforeSave, BeforePrint и BeforeClose. Кроме этого, в Excel 97 предоставляется поддержка элементов управления ActiveX, которые могут быть встроены в листы и диалоговые окна UserForm. Элементы управления ActiveX могут реагировать на широкий диапазон событий, например, GotFocus, MouseMove и DblClick.

Редактор VBE предоставляет пользователям значительно большую поддержку, чем в предыдущей версии редактора. Например, при написании кода появляются всплывающие подсказки со списком методов и свойств объекта, а также аргументами и значениями параметров для функций и методов. Утилита Object Browser (Просмотр объектов) теперь работает значительно лучше, поддерживая поиск и предоставляя подробную информацию о встроенных константах.

Компания Microsoft предоставила библиотеку расширений Extensibility, которая обеспечивает создание кода VBA, управляющего средой редактора VBE и проектов VBA. Это позволяет создавать код, получающий непосредственный доступ к модулям кода и диалоговым окнам UserForm. Существует возможность создавать приложения, которые выравнивают строки кода в модуле или экспортируют код из модулей в текстовые файлы.

Excel 97 была портирована на Macintosh и получил название Excel 98. К сожалению, большинство упрощавших работу программистов возможностей справочного руководства VBE не были включены в версию продукта на другой платформе. Возможности расширения редактора VBE также не были перенесены на платформу Macintosh.

Excel 2000

С точки зрения программирования на языке VBA в Excel 2000 значительных изменений не произошло. В этой версии Office и Excel был значительно переработан пользовательский интерфейс и улучшены некоторые возможности Excel, например PivotTable (сводная таблица). Была добавлена новая возможность PivotChart (сводная диаграмма). Изменения в Excel 2000 больше всего затронули пользователей Web. В основном это касалось возможности сохранения книг в виде Web-страниц. Кроме этого, появились новые возможности групповой работы, которые предоставляли механизмы распространения информации.

Одним из самых ожидаемых улучшений VBA было введение немодальных диалоговых окон UserForm. Ранее Excel поддерживала только модальные диалоговые окна, которые перехватывали фокус после появления на экране и не позволяли выполнять другие операции, пока окно не будет закрыто. Немодальные диалоговые окна могут использоваться для вывода заставок при загрузке приложения Excel или для вывода индикатора текущего состояния при выполнении большого макроса.

Excel 2002

В Excel 2002 также были предоставлены только минимальные улучшения. И в этот раз самые значительные улучшения касались пользовательского интерфейса, а не программных возможностей. Компания Microsoft продолжает концентрироваться на связанных с Web возможностях, которые значительно упрощают доступ к данным и их распространение через сеть Internet. Среди полезных для программистов VBA новых возможностей можно перечислить объекты `Protection`, `SmartTag`, `RTD` (данные реального времени) и расширенную поддержку XML.

Новый объект `Protection` позволяет избирательно управлять доступными для пользователя возможностями защищенной книги. Например, можно выборочно разрешить сортировку, изменение форматирования ячеек или вставку и удаление строк и столбцов. Кроме этого, предоставлен новый объект `AllowEditRange`, который может применяться для определения списка пользователей, имеющих право редактировать конкретные диапазоны (возможна защита с помощью пароля). К разным диапазонам могут применяться различные комбинации разрешений.

Смарт-теги позволяют Excel распознавать вводимые в ячейки данные, как имеющие специальное значение. Например, Excel 2002 может распознавать биржевые аббревиатуры (MSFT соответствует Microsoft Corporation). При обнаружении такой аббревиатуры Excel выводит символ смарт-тега, с которым связано всплывающее меню. Это меню можно использовать для получения дополнительной информации, например последней цены акции или итогового отчета компании. Компания Microsoft предоставляет инструментарий, позволяющий разработчикам создавать новое программное обеспечение смарт-тегов. В ближайшее время можно ожидать появления большого количества приложений, использующих возможности смарт-тегов для предоставления данных в пределах организации или в сети Internet.

Данные реального времени позволяют разработчикам создавать источники информации для пользователей. После создания ссылки на лист, изменения в данных передаются автоматически. Очевидным применением этой возможности является получение биржевых цен, которые меняются в реальном времени в процессе торгов. Еще одним приложением является возможность создания журнала показаний научных инструментов или промышленных контроллеров процессов. Как и в случае смарт-тегов, можно ожидать появления большого количества приложений, помогающих пользователям Excel получать доступ к динамической информации.

Расширенная поддержка XML означает более простое создание приложений, обменивающихся данными по сети Internet и корпоративным сетям. С усилением зависимости от развивающихся технологий эта возможность станет еще важнее.

Excel 2003

Web является неотъемлемой частью современной жизни. С увеличением значительности Web во всем мире компания Microsoft сместила внимание на предоставление решений, отвечающих возрастающей важности этих технологий в мире компьютеров. По этой причине в Excel 2003 можно обнаружить значительное количество изменений, отражающих перемены в современном мире.

Кроме новых, связанных с сетью Internet, возможностей, предоставляются расширенные возможности книг, новая функциональность для анализа данных, усовершенствованная поддержка XML и общего доступа к книгам из сети Internet, а также улучшенный внешний вид и поведение пользовательского интерфейса.

XML, или eXtensible Markup Language, является расширяемым языком разметки гипертекста. По сути, XML является текстовым открытым промышленным стандартом, поддерживающим расширение для различных применений и предназначенным для передачи информации в сети Internet. Увеличение поддержки XML упрощает обмен данными электронных таблиц Excel с другими промышленными решениями, которые используют XML.

Предоставляется расширенная поддержка управлением списками диапазонов, а пользовательский интерфейс улучшен через добавление возможностей для модификации, фильтрации и идентификации этих списков.

Упрощена процедура общего доступа и обновления данных Excel при использовании служб Windows Sharepoint Services. Например, изменения в списках Excel автоматически обновляются на сервере Sharepoint Services. Поддержка автономной модификации данных с их ресинхронизацией при следующем подключении упрощает работу пользователей, работающих за пределами сети, например, в ситуации Великого Затмения 2003*.

В Excel было добавлено несколько десятков новых и мощных статистических функций, а также возможность сравнения книг, больший объем справочной информации и поддержка связи с беспроводными устройствами, например, планшетными ПК. Все эти возможности позволяют сделать работу пользователей более продуктивной.

Что рассматривается в этой книге

В основном эта книга предназначена для пользователей Excel, которые желают совладать с мощью языка VBA в собственных приложениях Excel. Язык VBA всегда рассматривается в контексте Excel, а не только как язык программирования универсальных приложений.

Остальная часть книги делится на три раздела:

- программирование;
- расширенные возможности Excel;
- новые возможности, обеспечивающие интерактивный совместный доступ к информации.

В приложениях А, Б и В представлено полное справочное руководство по обновленной объектной модели.

Книга была реорганизована таким образом, чтобы информация о программировании была перенесена в начало. Кроме этого, были добавлены новые главы, в которых основное внимание уделяется объектно-ориентированному программированию, обработке ошибок и созданию стабильного кода. В этих главах предоставлена информация обо всем, от инкапсуляции, интерфейсов обработки ошибок и отладки до создания надстроек, программирования с использованием Windows API и решения проблем интернационализации.

В основном разделе книги рассматриваются новые расширенные возможности, доступные пользователям Excel. Эти возможности (например новая поддержка диапазонов

---

* *Отключение электроэнергии на всем восточном побережье Северной Америки в 2003 году.* — Примеч. ред.

и списков) необходимы для максимального использования потенциала Excel как средства программирования и средства повышения личной производительности.

В третьей части книги рассматриваются ресурсы для совместного доступа к данным Excel через сеть Internet.

Так как настоящая книга в основном является справочным руководством для программистов, основное внимание уделяется именно им. Но кроме этого, были оставлены главы, в которых описываются возможности, больше предназначенные для пользователей (при необходимости, содержимое таких глав было обновлено).

Номера версий

Изначально данная книга создавалась для Excel 2000 и теперь расширена для Excel 2003, входящей в состав Office XP. Учитывая, что изменения в объектной модели по сравнению с Excel 97 были минимальны, книга может использоваться для работы со всеми тремя версиями Excel. При обсуждении возможностей, которые не поддерживаются в более ранних версиях, на это будет явно указано в тексте.

Что нужно для использования книги

Практически для всех рассматриваемых тем приводятся примеры. Приводятся полные фрагменты кода, а также снимки экранов, если в этом возникает необходимость. Версия операционной системы Windows не имеет значения. Важно, чтобы на диск был установлен полный набор компонентов Excel, а при рассмотрении более сложных тем связи между Excel и другими приложениями Office потребуется установка полного состава Office. Удостоверьтесь, что установленные компоненты предоставляют доступ к редактору VBE и файлам справочного руководства по VBA. При установке эти компоненты можно отключить.

Обратите внимание, что в главах 13 и 14 требуется наличие установленной среды VB6, так как в этих главах рассматривается использование надстроек COM и смарт-тегов.

Соглашения

Для упрощения понимания текста в книге используется несколько соглашений по форматированию.

> В таких полях приводится важная информация, которую нужно запомнить. Эта информация непосредственно относится к тексту раздела.

Советы, подсказки, приемы и информация, не относящаяся непосредственно к рассматриваемой теме, выделяются таким шрифтом.

Стилевое оформление в тексте:

- важные слова выделяются при первом появлении в тексте;
- комбинации клавиш выделяются следующим образом <Ctrl+A>;
- имена файлов, URL и код в тексте выделяется таким шрифтом: `persistence.properties`;
- код выделяется двумя разными шрифтами.

```
В примерах новый и важный код выделяется серым фоном.
Для менее важного кода в данном контексте выделение серым фоном
не используется. Также такое выделение не используется для
повторяющегося кода.
```

Исходный код

При работе с примерами в данной книге можно вручную вводить весь код или воспользоваться файлами с исходным кодом, которые предоставляются вместе с книгой. Весь исходный код для этой книги доступен для загрузки на сайте `http://www.wrox.com`. После открытия сайта найдите раздел книги (воспользуйтесь полем Search или выберите название из списка) и щелкните на ссылке Download Code на странице, посвященной настоящей книге.

Так как многие книги имеют похожие названия, для поиска книги проще воспользоваться номером ISBN; эта книга имеет номер ISBN 0-764-55660-6.

После загрузки кода распакуйте его любым подходящим инструментом. Кроме этого, можно перейти на основную страницу загрузки кода издательства Wrox по адресу `http://www.wrox.com/dynamic/books/download.aspx` и просмотреть код, доступный для этой книги и других книг издательства Wrox.

Ошибки

Мы приложили все усилия, чтобы исправить ошибки в тексте и в коде. Но никто не совершенен, поэтому ошибки иногда встречаются. Если в одной из наших книг будет найдена ошибка (орфографическая или ошибка в коде), мы будем благодарны за сообщение о ней. Отправив сообщение об ошибке, вы сохраните другим читателям несколько часов неудачных попыток заставить код работать. В то же время, этим вы поможете нам выпускать более качественные книги.

Для доступа на страницу со списком ошибок в этой книге перейдите на сайт по адресу `http://www.wrox.com` и найдите книгу по названию с помощью поля Поиск (Search). После этого щелкните на ссылке Book Errata. На этой странице можно просмотреть все сообщения об ошибках, предоставленные пользователями и редакторами в издательстве Wrox. Полный список книг со ссылками на страницы сообщений об ошибках доступен по адресу `www.wrox.com/misc-pages/booklist.shtml`.

Если "ваша" ошибка отсутствует на странице со списком ошибок, переходите на страницу по адресу `http://www.wrox.com/contact/techsupport.shtml` и заполните форму для отправки нам сообщения об ошибке. Мы проверим полученную информацию и, в случае необходимости, опубликуем сообщение на странице сообщений об ошибках в этой книге. Обнаруженная проблема будет решена в последующих редакциях этой книги.

В случае кризиса

Существует множество мест, куда можно обратиться в случае проблем. Лучшим источником информации о всех аспектах Excel являются такие же пользователи. Они доступны во множестве конференций новостей в сети Internet. Попытайтесь настроить программу чтения конференций новостей на сайт, где достаточно людей, готовых помочь решить проблему:

- `msnews.microsoft.com`

Подпишитесь на группу microsoft.public.excel.programming или на другую подходящую группу новостей. Ответ на отправленный вопрос обычно появляется в течение часа.

Стивен Буллен и Роб Боуви поддерживают очень полезные Web-сайты, на которых можно найти большой объем информации и файлы, доступные для бесплатной загрузки. Сайты доступны по следующим адресам:

- http://www.bmsltd.co.uk
- http://www.appspro.com

Еще один полезный сайт поддерживается Джоном Уокенбахом (John Walkenbach) по адресу:

- http://www.j-walk.com

С издательством Wrox можно связаться непосредственно на сайтах:

- http://www.wrox.com — поддержка и загрузка исходного кода;
- http://p2p.wrox.com/list.asp?list=vba_excel — открытое обсуждение Excel VBA.

Непосредственные вопросы можно задавать по адресу pkimmel@softconcepts.com.

(Помните, что мы можем попытаться ответить на все вопросы, но нельзя ответить на все вопросы сразу. Желательно отправлять вопрос нескольким источникам информации, что позволит получить ответ быстрее и выбрать наиболее подходящее решение из нескольких ответов.)

Другие, связанные с компанией Microsoft, источники информации доступны по следующим адресам:

- http://www.microsoft.com/office/ — актуальные новости и поддержка;
- http://msdn.microsoft.com/office/ — новости для разработчиков и статьи о работе с продуктами Microsoft;
- http://www.microsoft.com/technet — статьи Microsoft Knowledge Base, информация о безопасности и большой объем другой информации, связанной с администрированием.

p2p.wrox.com

Для обсуждения с авторами и другими читателями присоединяйтесь к форумам P2P по адресу p2p.wrox.com. Форумы представляют собой основанную на Web систему отправки сообщений о книгах издательства Wrox и связанных с ними технологиях, а также взаимодействия с другими читателями и пользователями этих технологий. Форум предоставляет возможность подписки для получения интересующих тем по электронной почте. На этих форумах доступны авторы и редакторы из издательства Wrox, а также эксперты и читатели.

По адресу http://p2p.wrox.com доступно несколько форумов, которые могут помочь не только в процессе чтения книги, но и при разработке собственных приложений. Для присоединения к форумам необходимо выполнить такую последовательность действий.

1. Перейдите на сайт по адресу p2p.wrox.com и щелкните на ссылке Register (Регистрация).
2. Прочитайте правила пользования форумом и щелкните на кнопке Agree (Согласен).
3. Введите необходимую информацию, а также необязательную информацию, которую можно предоставить, и щелкните на кнопке Submit (Зарегистрировать).
4. По электронной почте будет отправлено сообщение с инструкциями по проверке учетной записи и завершении процесса присоединения к форумам.

Для чтения сообщений на форумах P2P регистрация не требуется, но для отправки собственных сообщений она обязательна.

После присоединения к форумам можно публиковать новые сообщения и отвечать на сообщения других пользователей. В Web сообщения можно читать в любой момент. Если необходимо, чтобы новые сообщения из определенного форума отправлялись по электронной почте, щелкните на кнопке Subscribe to this Forum (Подписка на этот форум) возле имени форума в списке форумов.

Дополнительная информация об использовании форума Wrox P2P доступна в списке часто задаваемых вопросов о работе программного обеспечения форума, а также о форумах P2P и книгах Wrox. Для чтения списка часто задаваемых вопросов щелкните на ссылке FAQ на любой странице сайта P2P.

Ждем ваших отзывов!

Вы, читатель этой книги, и есть главный ее критик и комментатор. Мы ценим ваше мнение и хотим знать, что было сделано нами правильно, что можно было сделать лучше и что еще вы хотели бы увидеть изданным нами. Нам интересно услышать и любые другие замечания, которые вам хотелось бы высказать в наш адрес.

Мы ждем ваших комментариев и надеемся на них. Вы можете прислать нам бумажное или электронное письмо либо просто посетить наш Web-сервер и оставить свои замечания там. Одним словом, любым удобным для вас способом дайте нам знать, нравится вам эта книга или нет, а также выскажите свое мнение о том, как сделать наши книги более интересными для вас.

Посылая письмо или сообщение, не забудьте указать название книги и ее авторов, а также ваш обратный адрес. Мы внимательно ознакомимся с вашим мнением и обязательно учтем его при отборе и подготовке к изданию последующих книг. Наши координаты:

E-mail: info@dialektika.com
WWW: http://www.dialektika.com

Адреса для писем:

из России: 115419, Москва, а/я 783
из Украины: 03150, Киев, а/я 152

1

Пример использования VBA в Excel

Эта глава предназначена для тех, кто не знаком с Excel и записью макросов в Excel, а также тех, кто не обладает достаточным опытом программирования с помощью языка Visual Basic for Applications (VBA). Если предоставляемые Excel возможности уже известны, запись макросов не вызывает трудностей и есть опыт применения языка VBA и редактора VBE, можно сразу переходить к главе 3.

В этой же главе предоставляется информация, которая необходима для удобного перехода к расширенным возможностям, представленным в следующих главах. Здесь рассматриваются следующие темы:

- запись макросов в Excel;
- определяемые пользователем функции;
- объектная модель Excel;
- концепции программирования с использованием языка VBA.

Язык Excel VBA является языком программирования, позволяющим использовать код Visual Basic для управления многими возможностями пакета Excel. VBA позволяет модифицировать поведение приложений Excel. Обычно фрагменты кода VBA называются макросами. В этой главе рассматривается более формальная терминология, но термин макрос будет использоваться как универсальное название для фрагмента кода VBA.

Если при обычном использовании Excel определенная последовательность команд выполняется несколько раз, выполнение этих операций из макроса позволяет заметно сэкономить усилия и время. Если приложение Excel необходимо предоставить пользователям, не знакомым с принципами работы в Excel, с помощью макросов можно создать

кнопки и диалоговые окна, которые проведут через приложение и позволят автоматизировать промежуточные операции.

Если операцию можно выполнить вручную, запись макроса позволит получить последовательность команд для выполнения этой операции. Это быстрый и простой процесс, не требующий предварительных знаний VBA. Многие пользователи Excel записывают и запускают макросы, не утруждая себя изучением языка VBA.

Но полученный в результате записи макроса код может оказаться недостаточно гибким, то есть, данный конкретный макрос может быть использован для выполнения определенной операции на определенном диапазоне ячеек. Кроме этого, записанный макрос может работать значительно медленнее, чем код, написанный знакомым с языком VBA программистом. Если необходимо создавать интерактивные макросы, адаптирующиеся к изменениям и быстро выполняющие свою функцию, а также макросы, которые используют такие расширенные возможности Excel, как собственные диалоговые окна, язык VBA придется изучить.

> Не подумайте, что мы не рекомендуем записывать макросы. Запись макросов является наиболее ценным инструментом программистов на языке VBA. Это самый быстрый способ генерации работающего кода. Но для получения гибкого и эффективного кода к записанному макросу необходимо приложить собственное знание языка VBA. В этой книге часто рассматриваются ситуации, когда сначала записывается макрос, после чего выполняется адаптация кода макроса.

В данной главе будет показано, как записывать и запускать макросы в Excel, а также как использовать редактор VBE для просмотра и изменения кода макроса, выходя за пределы записи макросов и прикасаясь к мощи языка VBA и объектной модели Excel.

Кроме этого, язык VBA может использоваться для создания собственных функций листов. В составе Excel предоставляются сотни встроенных функций, например СУММ и ЕСЛИ, которые можно применять в формулах внутри ячеек. Но если часто используется сложный метод расчета, который не входит в список стандартных функций Excel (например, расчет налогов или специализированная формула), можно определить собственную функцию.

Использование механизма записи макросов

Запись макросов в Excel выполняется так же, как запись приветствия на автоответчике. Для записи приветствия сначала его необходимо отрепетировать. После этого включить автоответчик и записать приветствие. После завершения можно отключить автоответчик. Приветствие будет звучать каждый раз, когда никто не берет трубку телефона.

Запись макроса в Excel происходит подобным образом. Сначала нужно продумать необходимые операции и решить, в какие моменты начнется и закончится запись макроса. Для этого необходимо подготовить электронную таблицу, включить механизм записи макроса, выполнить определенную последовательность операций и отключить механизм записи макроса. В результате будет получена автоматизированная процедура, которую можно будет запустить одним нажатием кнопки.

Запись макроса

Предположим, что необходимо создать макрос, который выводит шесть названий месяцев в виде трехбуквенных аббревиатур в верхней строке листа начиная с ячейки B1. Это достаточно простой макрос, так как этого же эффекта можно добиться с помощью

операции автоматического заполнения (AutoFill), но данный пример будет использован для демонстрации важных универсальных концепций.

- Сначала представьте, как будет выполняться эта операция. В данном случае все просто — необходимо ввести данные в лист. Помните, что более сложный макрос может потребовать более длительного обдумывания перед записью.
- После этого подумайте, в какой момент необходимо начать запись. В данном случае в макрос необходимо включить операцию выделения ячейки B1. Таким образом, название месяца "Янв" всегда будет попадать в ячейку B1. Если операцию выделения не включить в макрос, будет записан ввод последовательности символов "Янв" в активную ячейку. При воспроизведении макроса активная ячейка может находиться в любом месте.
- После этого подумайте, в какой момент необходимо завершить запись. Может возникнуть желание включить в состав макроса операцию форматирования, например выделение содержимого ячеек полужирным шрифтом и курсивом. Кроме этого, необходимо выбрать положение активной ячейки после завершения работы макроса. Это может быть как ячейка со словом "Июн", так и ячейка в следующей строке столбца A или столбца B, что позволит сразу же начать ввод данных. Предположим, что в качестве активной ячейки после завершения работы макроса выбрана ячейка A2, поэтому перед отключением механизма записи макроса необходимо выделить ячейку A2.
- Теперь можно настроить экран и начинать запись макроса.

В данном случае можно начинать с пустого листа, на котором выделена ячейка A1. Если отдается предпочтение работе с панелями инструментов, воспользуйтесь командой Вид⇨Панели инструментов (View⇨Toolbars) и установите флажок напротив Visual Basic. Для начала записи макроса нажмите кнопку Записать макрос (Record Macro) (на кнопке изображена красная точка). При желании, запись макроса можно включить с помощью команды Сервис⇨Макрос⇨Записать макрос (Tools⇨Macro⇨Record New Macro) из панели меню листа.

В поле Макрос (Macro name) замените принятое по умолчанию значение Macro1 на имя, которое будет присвоено записанному макросу. Имя должно начинаться с буквы и содержать только буквы, цифры и символ подчеркивания. Максимальная длина имени составляет 255 символов. Имя макроса не должно содержать специальных символов, например, !, ? или пробелов. Кроме этого, лучше использовать короткие описательные имена, легко узнаваемые в будущем. Для разделения слов можно применять символ подчеркивания, но намного лучше для этих целей подходит использование заглавных букв.

> Обратите внимание, что разделение слов в имени переменной с помощью заглавных букв называется разделением в стиле Паскаль, как в РазделениеВСтилеПаскаль. Использование прописной буквы в начале первого слова и заглавных букв в начале всех остальных слов называется разделением в стиле верблюда, как в разделениеВСтилеВерблюда*. Такие чувствительные к регистру языки, как C++, Pascal и C#, предлагают варианты в написании заглавных букв в виде соглашения по именованию. Многие программисты следуют этим соглашениям. Так как VBA не чувствителен к регистру, можно использовать любой удобный стандарт.

* *Есть горбы и начинается с длинной шеи.* — Примеч. ред.

Назовите макрос **MonthNames1**, так как скоро будет создана еще одна версия макроса.

В поле Сочетание клавиш (Shortcut key:) введите единственный символ. Позднее нажатие этой клавиши в комбинации с клавишей <Ctrl> приведет к запуску макроса. В данном случае будет использован символ m. Точно так же можно использовать символ M. В таком случае для запуска макроса при нажатии клавиши <M> придется удерживать клавиши <Ctrl> и <Shift>. Комбинация клавиш необязательна. Макрос можно запустить и несколькими другими способами.

В поле Описание (Description:) можно оставить предложенное описание или ввести собственный комментарий. Эта последовательность символов будет указана в начале кода макроса. Она не имеет никакого значения для интерпретатора VBA, но предоставляет вам и другим пользователям информацию о макросе. Комментарии можно отредактировать позднее, поэтому сейчас лучше оставить принятое по умолчанию описание. Все макросы Excel создаются в пределах книги.

В поле Сохранить в (Store macro in:) можно выбрать, где будет сохранен записанный макрос. Предоставляется три варианта. Если выбрать Новая книга (New Workbook), для макроса будет открыта новая пустая книга. Вариант Личная книга макросов (Personal Macro Workbook) соответствует специальной скрытой книге, которая рассматривается ниже. В данном случае будет выбран пункт Эта книга (This Workbook), чтобы макрос был сохранен в активной книге (рис. 1.1).

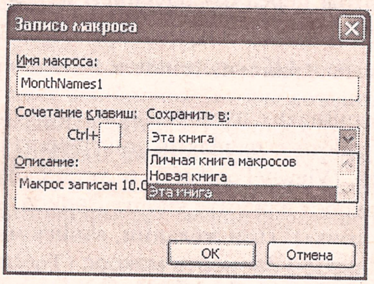
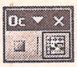


Рис. 1.1. Выбор книги для сохранения макроса

После заполнения полей в диалоговом окне Запись макроса (Record Macro) щелкните на кнопке OK. Слева в строке состояния в нижней части окна появится надпись Запись (Recording), а на экране появится панель инструментов Остановить запись (Stop Recording). Обратите внимание, что если во время предыдущей попытки записи макроса панель Остановить запись (Stop Recording) была закрыта, она на экране не появляется. Если панель не появилась, восстановите ее в соответствии с инструкциями в разделе "Абсолютная и относительная запись". Но в данном случае эта панель не нужна, так как запись можно остановить из панели инструментов Visual Basic или из меню Сервис (Tools).

Если панель Остановить запись (Stop Recording) появилась на экране, удостоверьтесь, что кнопка Относительная ссылка (Relative Reference) отключена. Вокруг кнопки должна отсутствовать рамка, то есть кнопка должна выглядеть не так, как показано на рис. 1.2. По умолчанию при записи макросов используются абсолютные ссылки на ячейки.

Рис. 1.2. Включение относительных ссылок при записи макроса



После этого необходимо щелкнуть на ячейке B1, ввести "Янв" и заполнить остальные ячейки, как показано на рис. 1.3. Теперь выделите ячейки B1:G1 и щелкните на кнопках Полужирный (Bold) и Курсив (Italic) на панели инструментов Форматирование (Formatting). Щелкните на ячейке A2 и остановите запись макроса.

Для остановки записи макроса можно щелкнуть на кнопке Остановить запись (Stop Recording) на панели инструментов Остановить запись (Stop Recording) или щелкнуть на кнопке Остановить запись (Stop Recording) на панели управления Visual Basic (после начала записи кнопка Начать запись (Start Recording) меняется на кнопку Остановить запись (Stop Recording)). Кроме этого, можно воспользоваться командой меню Сервис⇨Макрос⇨Остановить запись (Tools⇨Macro⇨Stop Recording). Сохраните книгу под именем `Recorder.xls`.

> Важно не забыть запись макроса. Если оставить механизм записи включенным и попытаться запустить записанный макрос, приложение может войти в цикл, в котором макрос постоянно запускает сам себя. В таком случае (а также в случае появления других ошибок) удерживайте клавишу <Ctrl> и нажмите клавишу <Break> для немедленного завершения работы макроса. После этого можно завершить выполнение макроса или переключиться в режим отладки для поиска ошибок. Кроме этого, работу макроса можно прервать с помощью нажатия клавиши <Esc>, но этот способ работает не так эффективно, как комбинация <Ctrl+Break>.

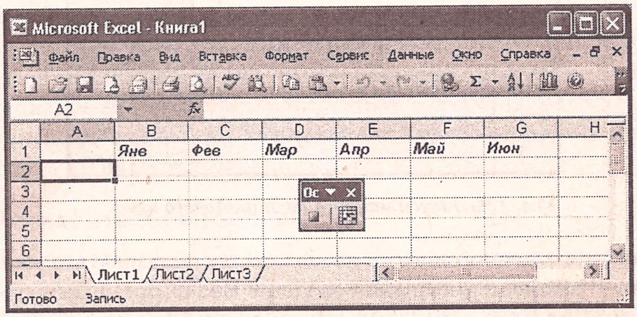


Рис. 1.3. Ввод значений в процессе записи макроса

Личная книга макросов

Если выбрано сохранение макроса в Личной книге макросов, макрос добавляется в специальный файл, который называется `Personal.xls`. Это скрытый файл, сохраняемый в каталоге Excel `Startup` при завершении работы Excel. Это значит, что файл `Personal.xls` автоматически загружается при запуске Excel и сохраненные в нем макросы доступны для всех остальных книг.

Если файл `Personal.xls` еще не существует, он будет создан механизмом записи макросов. Для просмотра этой книги в окне Excel можно воспользоваться командой меню Окно⇨Отобразить (Window⇨Unhide), но эта возможность редко оказывается востребованной, так как для редактирования макросов из книги `Personal.xls` лучше воспользоваться редактором VBE. Как исключение, книгу `Personal.xls` можно сделать видимой для сохранения данных на листах этой книги. Для сокрытия книги после сохранения

данных можно воспользоваться командой меню Окно⇨Скрыть (Window⇨Hide). Если создается универсальный вспомогательный макрос, который должен быть доступен в любой книге, сохраните этот макрос в книге `Personal.xls`. Если макрос относится к приложению в текущей книге, сохраните макрос вместе с приложением.

Запуск макросов

Для запуска макроса вставьте новый лист в книгу `Recorder.xls` или откройте новую пустую книгу, оставив в памяти книгу `Recorder.xls`. Макросы можно запускать только из открытой книги, но работать они могут в любой другой открытой книге.

Для запуска макроса удерживайте клавишу <Ctrl> и нажмите клавишу <m>, которая была назначена макросу в процессе записи. Кроме этого, для запуска макроса можно воспользоваться командой меню Сервис⇨Макрос⇨Макросы (Tools⇨Macro⇨Macros) и выполнить двойной щелчок на имени макроса. Также можно выбрать имя макроса в списке и щелкнуть на кнопке Выполнить (Run), как показано на рис. 1.4.

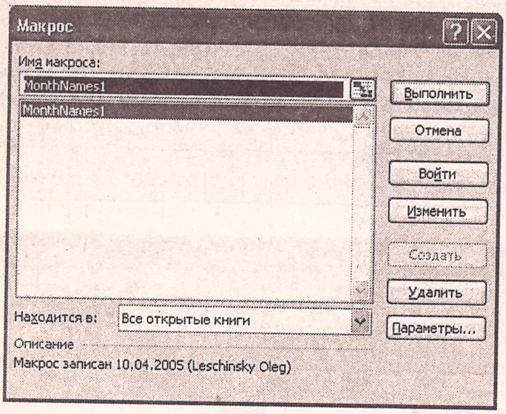


Рис. 1.4. Запуск макроса из диалогового окна

Для открытия этого диалогового окна можно воспользоваться кнопкой Выполнить макрос (Run Macro) в панели инструментов Visual Basic (рис. 1.5).

Рис. 1.5. Запуск макроса из панели инструментов

Комбинации клавиш

Для изменения назначенной макросу комбинации клавиш откройте диалоговое окно Макрос (Macro) (для этого выберите Сервис⇨Макрос⇨Макросы (Tools⇨Macro⇨Macros) или щелкните на кнопке Выполнить макрос (Run Macro) на панели инструментов Visual Basic). После этого выберите имя макроса из списка и щелкните на кнопке Параметры (Options). Откроется диалоговое окно, показанное на рис. 1.6.

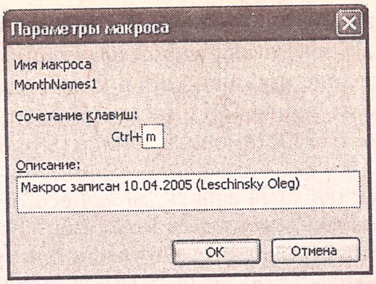


Рис. 1.6. Настройка параметров макроса

С помощью этого диалогового окна макросам можно назначить одну и ту же комбинацию клавиш в пределах одной книги (диалоговое окно, которое появляется при записи макроса, не позволяет задать макросу уже назначенную комбинацию клавиш).

> Кроме этого, высока вероятность, что макросы с одинаковыми комбинациями клавиш будут находиться в разных открытых книгах. Возникает проблема определения макроса, который будет запущен при нажатии конкретной комбинации клавиш. Для этого существует правило, по которому запускается тот макрос, имя которого оказывается первым в алфавитном порядке.

Комбинации клавиш имеют смысл для тех макросов, которые используются очень часто, особенно, если большую часть времени руки находятся на клавиатуре. Стоит запомнить комбинации клавиш, используемые регулярно. Для макросов, которые запускаются редко или предназначены для облегчения работы менее опытных пользователей, комбинации клавиш не нужны. Лучше назначить таким макросам осмысленные имена и запускать их из диалогового окна Макрос. С другой стороны, макросы можно запускать с помощью кнопок, размещенных на листе или на панели инструментов. Ниже будет показано, как добавлять такие кнопки.

Абсолютная и относительная запись

При запуске макроса `MonthNames1` он возвращается к тем же ячейкам, которые были выбраны при вводе имен месяцев. Не важно, какая ячейка была активна на момент запуска макроса. Если в макросе присутствует команда выделения ячейки B1, именно эта ячейка и будет выделена. Макрос выбирает ячейку B1, так как он создавался с использованием абсолютных ссылок. Альтернативный режим относительных ссылок запоминает положение ячеек относительно активной ячейки. Если выделить ячейку A10, включить запись макроса и выделить ячейку B10, будет записан выбор ячейки справа от активной, а не ячейки B10.

Ниже будет записан второй макрос, который называется `MonthNames2`. В этом макросе есть три отличия от предыдущего:

- в панели инструментов Остановить запись (Stop Recording) будет нажата кнопка Относительная ссылка (Relative Reference). Это первая операция после включения механизма записи макроса;
- перед вводом не будет выбираться ячейка “Янв”. При запуске макроса слово “Янв” должно вводиться в активную ячейку;
- перед завершением записи макроса будет выбрана ячейка под словом “Янв”, а не ячейка A2.

Начните с пустого листа и выделите ячейку B1. Включите запись макроса и введите имя макроса MonthNames2. В поле назначенной клавиши введите букву **M** (верхний регистр). Повторное использование клавиши <m> в качестве комбинации для запуска макроса невозможно. Щелкните на кнопке OK и на кнопке Относительная ссылка (Relative Reference) в панели инструментов Остановить запись (Stop Recording).

> Если панель инструментов Остановить запись (Stop Recording) не появляется автоматически после начала записи макроса, выберите в меню Вид⇨Панели инструментов (View⇨Toolbars) и установите флажок Остановить запись (Stop Recording). После этого панель инструментов Остановить запись (Stop Recording) должна появиться на экране. Обратите внимание, что после этого нужно немедленно щелкнуть на кнопке Остановить запись (Stop Recording) и начать запись еще раз. В противном случае записанный макрос будет выводить на экран панель инструментов при каждом запуске. После этого панель инструментов будет появляться при каждом запуске механизма записи макросов, если не закрыть ее в процессе записи.

Если возникла необходимость ресинхронизации панели инструментов Остановить запись (Stop Recording) в соответствии с приведенными ранее инструкциями, клавиша <M> уже будет назначена. Если не удается назначить макросу MonthNames2 клавишу <M>, воспользуйтесь другой клавишей, например <N>. После завершения записи назначенную комбинацию клавиш можно будет изменить. Для этого воспользуйтесь командой меню Сервис⇨Макрос⇨Макросы (Tools⇨Macro⇨Macros), выберите имя макроса и щелкните на кнопке Параметры (Options), как было показано ранее в разделе "Комбинации клавиш".

Введите **"Янв"** и названия других месяцев, как и при записи макроса MonthNames1. Выделите ячейки B1:G1 и щелкните на кнопках Полужирный (Bold) и Курсив (Italic) на панели инструментов Форматирование (Formatting).

> Удостоверьтесь, что диапазон B1:G1 выделяется слева направо, и ячейка B1 остается активной. В процессе записи макроса существует небольшая особенность, которая может привести к появлению ошибок при выделении ячеек справа налево или снизу вверх. При записи макроса с относительными ссылками всегда выделяйте диапазон начиная с верхнего левого угла. Эта проблема характерна для всех версий Excel VBA. (Выбор ячеек справа налево приводит к появлению ошибки времени выполнения 1004 при попытке запуска такого макроса.)

Наконец, выберите ячейку B2 под ячейкой со словом "Янв" и остановите запись макроса.

Перед запуском макроса MonthNames2 выберите начальную ячейку, например, A10. В этот раз макрос будет вводить имена месяцев в строке 10 начиная со столбца A. В завершение своей работы макрос выделит ячейку под начальной ячейкой.

Перед записью макроса, выделяющего ячейки, необходимо обдумать причины выбора абсолютных или относительных ссылок. Если ячейки выбираются для ввода данных или указания области печати, стоит использовать абсолютные ссылки. Если необходимо запускать макрос в различных областях листа, стоит использовать относительные ссылки.

Если для выбора последней ячейки массива данных необходимо воспроизвести поведение комбинации клавиш <Ctrl+клавиша управления курсором>, стоит воспользоваться относительными ссылками. Между относительными и абсолютными ссылками можно переключаться непосредственно в процессе записи макроса. Может потребоваться выделение вершины столбца с применением абсолютной ссылки, после чего переключиться на использование относительных ссылок и воспользоваться комбинацией клавиш <Ctrl+вниз> для выбора последней ячейки столбца и клавишей <вниз> для выбора следующей свободной ячейки.

Впервые возможность записи выбора блока ячеек с помощью комбинации клавиши <Ctrl> и клавиш управления курсором появилась в Excel 2000. Если выделить верхний левый угол массива данных, то, удерживая комбинацию <Ctrl+Shift> и нажав <↓> и <→>, можно выделить весь блок данных (если в блоке отсутствуют промежутки). Если записать эти операции с применением относительных ссылок, макрос можно использовать для выбора блока переменной размерности. В предыдущих версиях Excel записывался выбор блока исходного размера с применением абсолютных ссылок вне зависимости от режима записи.

Редактор VBE

Теперь пришло время познакомиться с процессами, которые происходят за кулисами. Если необходимо понять, что представляют собой макросы, попытаться модифицировать макрос и прикоснуться к полной мощи языка VBA, придется научиться пользоваться редактором VBE, который работает в собственном окне, отдельно от окна Excel. Пользователю предоставляется несколько методов активизации редактора VBE.

Редактор можно запустить с помощью кнопки Редактор Visual Basic (Visual Basic Editor) на панели инструментов Visual Basic. Кроме этого, для активизации редактора можно нажать и удерживать клавишу <Alt>, после чего нажать клавишу <F11>. Комбинация <Alt+F11> выступает в роли переключателя, позволяющего переходить между окнами Excel и VBA. Если необходимо отредактировать конкретный макрос, воспользуйтесь командой меню Сервис⇨Макрос⇨Макросы (Tools⇨Macro⇨Macros) для открытия окна Макросы (Macros), выберите макрос и щелкните на кнопке Изменить (Edit). Внешний вид окна VBE показан на рис. 1.7.

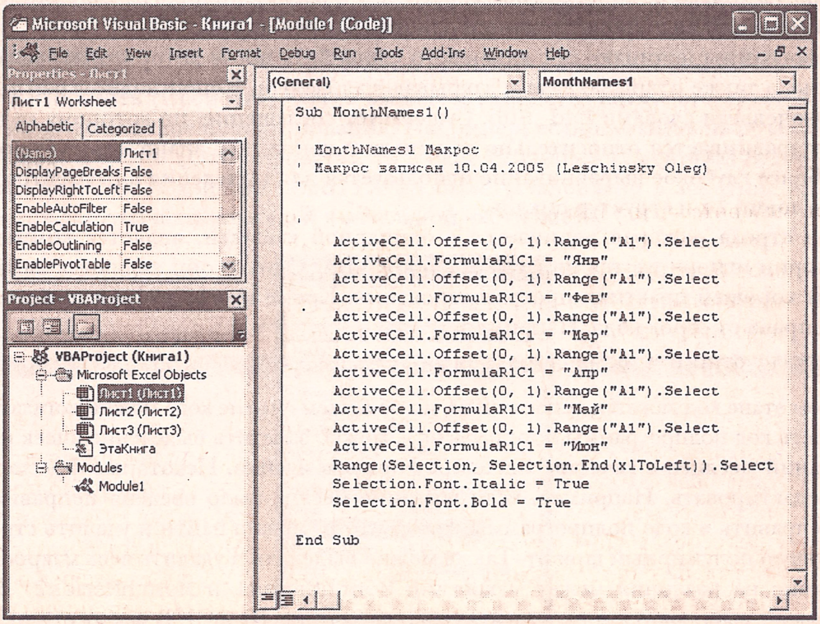


Рис. 1.7. Записанный макрос в окне редактора VBE

Вполне вероятно, что после первого запуска редактора VBE на экране будет видна только панель меню. Если панели инструментов отсутствуют, выберите View⇨Toolbars *(Вид⇨Панели инструментов) и установите флажок* Standard *(Стандартная). Выберите* View⇨Project Explorer *(Вид⇨Окно проекта) и* View⇨Properties *(Вид⇨Свойства) для отображения показанных слева окон. Если модуль кода справа отсутствует, выполните двойной щелчок на пиктограмме* Module1 *в окне* Project Explorer *(Окно проекта).*

Модули кода

Все макросы расположены в модулях кода. Один из модулей показан справа в окне VBE на рис. 1.7. Существует два типа модулей кода — стандартные и модули классов. На рис. 1.7 показан стандартный модуль. Модуль класса можно использовать для создания собственных объектов. Дополнительная информация о работе с модулями классов приводится в главе 6.

Некоторые модули классов настроены предварительно. Они связаны с листами в книге, а один модуль класса связан со всей книгой. Эти модули видны в окне Project Explorer (Окно проекта) в папке `Microsoft Excel Objects`. Дополнительная информация об этих модулях приводится далее в данной главе.

В книгу можно добавлять любое количество модулей. Показанный выше модуль `Module1` был добавлен механизмом записи макросов. В пределах каждого модуля может содержаться несколько макросов. В небольших приложениях все макросы обычно стоит хранить в пределах одного модуля. В больших проектах для организации кода не связанные друг с другом макросы можно хранить в разных модулях.

Процедуры

В VBA макросы называются процедурами. Существует два типа процедур — подпрограммы и функции. Функции рассматриваются в следующем разделе. При записи макросов создаются только подпрограммы. На рис. 1.7 показана подпрограмма `MonthNames1`, которая была создана в процессе записи одноименного макроса.

Подпрограммы начинаются с ключевого слова `Sub`, после которого указывается имя процедуры, а также открывающая и закрывающая скобки. Конец подпрограммы обозначается ключевыми словами `End Sub`. Существует соглашение, по которому код подпрограммы выравнивается относительно ее начала и конца, что значительно упрощает его чтение. Более глубокое выравнивание используется для выделения фрагментов кода, например условий `If` и структур циклов.

Любая строка, которая начинается с одинарной кавычки, является комментарием. Комментарии игнорируются интерпретатором VBA. Однако они являются важным компонентом хорошей практики программирования. Кроме этого, комментарии можно добавлять справа от строк кода. Например:

```
Range("B1").Select ' Выделение ячейки B1
```

На этом этапе код кажется непонятным, но об общем смысле кода можно догадаться. Если просмотреть код подпрограммы `MonthNames1`, можно заметить выделение ячеек и присвоение названий месяцев в качестве значения формулы ячейки. Некоторые фрагменты кода можно редактировать. Например, если название месяца было введено неправильно, его можно исправить в коде подпрограммы; кроме этого, можно найти и удалить строку, устанавливающую полужирный шрифт. Также можно выделить и удалить весь макрос. Обратите внимание на различия между макросами `MonthNames1` и `MonthNames2`. В макросе `MonthNames1` выбираются конкретные ячейки, например, B1 и C1. В макросе `MonthNames2` для выбора ячеек в этой же строке справа от активной ячейки используется ключевое слово `Offset`. Уже сейчас стоит сказать, что возможности языка VBA стали доступнее.

Окно проекта

Окно Project Explorer (Окно проекта) является инструментом навигации. Каждая книга содержит собственный проект VBA. В окне Project Explorer (Окно проекта) выводится список всех открытых проектов, а также их компонентов (рис. 1.8).

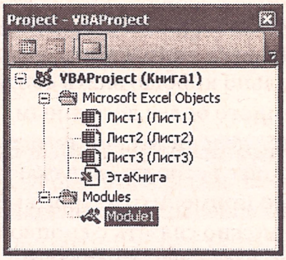


Рис. 1.8. Окно проекта

Окно Project Explorer (Окно проекта) можно использовать для поиска и активизации модулей кода в пределах проекта. Для открытия и активизации модуля достаточно выполнить двойной щелчок на пиктограмме модуля. Кроме этого, в окне Project Explorer (Окно проекта) можно вставлять и удалять модули кода. Щелкните правой кнопкой мыши в любом месте окна Project Explorer (Окно проекта) и выберите меню Insert (Вставка) для добавления нового стандартного модуля, модуля класса или диалогового окна `UserForm`.

Для удаления модуля `Module1` щелкните правой кнопкой мыши на пиктограмме модуля и выберите пункт Remove Module (Удалить модуль). Обратите внимание, что для модулей, связанных с книгой и листом, эта операция недоступна. Кроме этого, код из модуля можно экспортировать в отдельный текстовый файл или импортировать из текстового файла.

Окно Свойства

В окне Properties (Свойства) всегда выводятся свойства активного объекта, которые можно изменить на этапе проектирования. Например, если щелкнуть на объекте `Sheet1` в окне Project Explorer (Окно проекта), то в окне Properties (Свойства) будут показаны следующие свойства. Свойство `ScrollArea` было установлено в значение A1:D10 для ограничения доступа пользователей к остальной части листа.

Рис. 1.9. Установка свойства объекта

Поддерживается быстрый переход от свойства к соответствующей странице справочного руководства. Достаточно выделить свойство, например `ScrollArea`, показанное на рис. 1.9, и нажать клавишу <F1>.

Другие методы запуска макросов

Ранее было показано, как запускать макросы с помощью комбинации клавиш или из меню Сервис (Tools). Ни один из этих методов не оказался достаточно удобным. Для их использования необходимо довольно хорошо знать собственные макросы. Связав макрос с кнопкой, его можно сделать намного более доступным.

Если макрос относится к определенному листу и будет использоваться в пределах конкретного диапазона листа, стоит встроить запускающую кнопку возле соответствующего диапазона. Если необходимо применить макрос вместе с любым листом или книгой в любом диапазоне листа, то его можно связать с кнопкой на панели инструментов.

Существует множество других объектов, с которыми можно связать макросы. Это касается списков, комбинированных списков, полос прокрутки, флажков и переключателей. Все эти объекты называются элементами управления. Дополнительная информация о них приводится в главе 10. Кроме этого, макросы можно связывать с графическими объектами на листе, например с геометрическими фигурами, созданными с помощью панели инструментов Рисование (Drawing).

Кнопки на листе

В Excel 2003 предоставляются два разных набора элементов управления, которые могут быть встроены в лист. Один набор доступен на панели инструментов Формы (Forms), а второй — в панели инструментов Элементы управления (Control Toolbox). Панель инструментов Формы (Forms) унаследована от Excel 5 и Excel 95. Эти элементы управления используются в диалоговых листах Excel 5 и Excel 95 для создания диалоговых окон. В Excel 97 появились новые элементы управления ActiveX, которые доступны на панели инструментов Элементы управления (Control Toolbox). Эти элементы управления можно использовать в диалоговых окнах `UserForm` в редакторе VBE для создания диалоговых окон.

Для сохранения совместимости с более старыми версиями Excel, в Excel 97 и более старых версиях поддерживаются оба набора элементов управления и методы создания диалоговых окон. Если сохранение обратной совместимости с Excel 95 и Excel 5 не требуется, можно использовать только элементы управления ActiveX (кроме встраивания элементов управления в диаграммы; на данный момент в диаграммы можно встраивать только элементы управления Формы (Forms)).

Панель инструментов Формы

Еще одной причиной использования элементов управления Формы (Forms) является простота их применения, так как они не обладают всеми возможностями элементов управления ActiveX. Например, элементы управления Формы (Forms) могут реагировать только на единственное предопределенное событие, которым обычно является щелчок мышью. Элементы управления ActiveX могут реагировать на большое количество событий, например на щелчок мышью, двойной щелчок и нажатие клавиши на клавиатуре. Если эти возможности не нужны, можно остановиться на использовании элементов управления Формы (Forms). Для отображения панели инструментов Формы (Forms) необходимо выбрать пункт Вид⇨Панели инструментов⇨Формы (View⇨Toolbars⇨Forms). Для создания кнопки на листе щелкните на четвертой слева кнопке на панели инструментов Формы (Forms) (рис. 1.10).

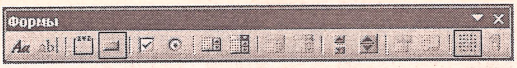


Рис. 1.10. Панель инструментов Формы

После этого кнопку можно перетащить на лист. Для этого нужно щелкнуть на листе в том месте, где должен находиться угол кнопки. Потом указатель необходимо перетащить в ту точку, в которой должен находиться противоположный угол кнопки. Откроется показанное на рис. 1.11 диалоговое окно, предоставляющее возможность связывания макроса с созданной кнопкой.

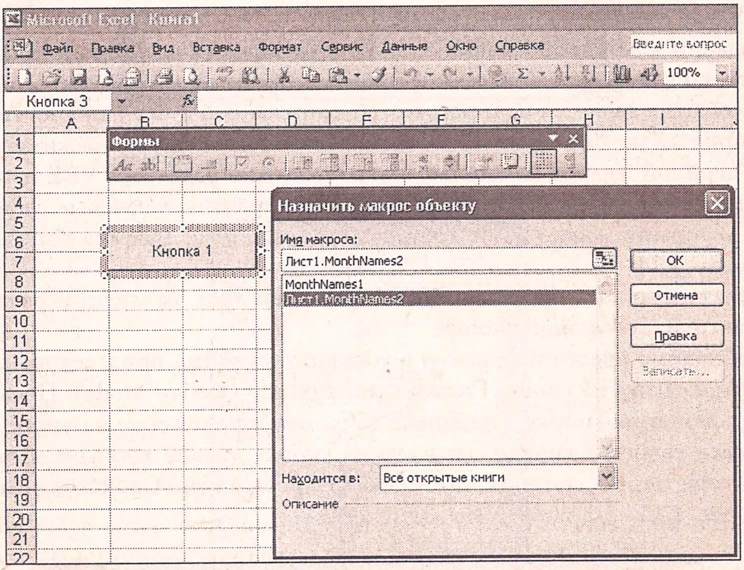


Рис. 1.11. Связывание макроса с кнопкой на листе

Щелкните на кнопке OK для завершения назначения. После этого можно отредактировать текст на кнопке, что позволит более ясно указать ее назначение. После щелчка на ячейке листа щелчок на кнопке приводит к запуску связанного с кнопкой макроса. Для редактирования кнопки щелкните на ней правой кнопкой мыши. Это приведет к выделению элемента управления и появлению контекстного меню. Если появление контекстного меню нежелательно, удерживайте клавишу <Ctrl> и для выделения кнопки щелкните на ней левой кнопкой мыши. (Не перетаскивайте указатель мыши при нажатой клавише <Ctrl>, так как это приведет к созданию копии кнопки.)

Если кнопку необходимо выровнять относительно направляющих линий на листе, нажмите и удерживайте клавишу <Alt> в процессе перетаскивания указателя мыши. Если размер кнопки уже указан, выделите кнопку и удерживайте нажатой клавишу <Alt>, растягивая кнопку за белые прямоугольники на краях кнопки. Перетаскиваемый край кнопки будет выровнен по ближайшей направляющей линии.

Панель инструментов Control Toolbox

Для создания кнопки на основе элемента управления ActiveX щелкните на шестой слева кнопке панели инструментов Элементы управления (Control Toolbox) (рис. 1.12).

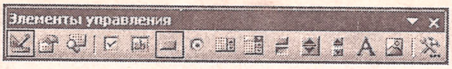


Рис. 1.12. Панель инструментов Элементы управления

При перетаскивании кнопки на лист приложение переходит в режим конструктора. В режиме конструктора щелчок левой кнопкой мыши позволяет выбрать элемент управления и начать редактирование. Для того чтобы элемент управления реагировал на события, режим конструктора нужно отключить. Для этого необходимо отключить пиктограмму режима конструктора на панели инструментов Элементы управления (Control Toolbox) или на панели управления Visual Basic, как показано на рис. 1.13.

При создании командной кнопки ActiveX возможность назначения макроса не предоставляется, но для кнопки необходимо написать процедуру обработки события щелчка. Процедура события является подпрограммой, запускаемой при возникновении события, например, при щелчке на кнопке. Для создания такой процедуры в режиме конструктора выполните двойной щелчок на кнопке. Откроется окно редактора VBE и будет показан модуль кода, связанный с листом. В модуль будут вставлены ключевые слова `Sub` и `End Sub`, между которыми можно будет добавить строки кода из макроса `MonthNames2`, как показано на рис. 1.14.

Для запуска этого кода переключитесь обратно на лист, отключите режим конструктора и щелкните на командной кнопке.

Если необходимо внести изменения в командную кнопку, придется включить режим конструктора, щелкнув на кнопке Режим конструктора (Design Mode). После этого можно выбрать командную кнопку и изменить ее размер и положение на листе. Кроме того, можно вывести свойства кнопки, щелкнув на кнопке правой кнопкой мыши и выбрав Свойства (Properties) из контекстного меню. Откроется окно Свойства (Properties), показанное на рис. 1.15.

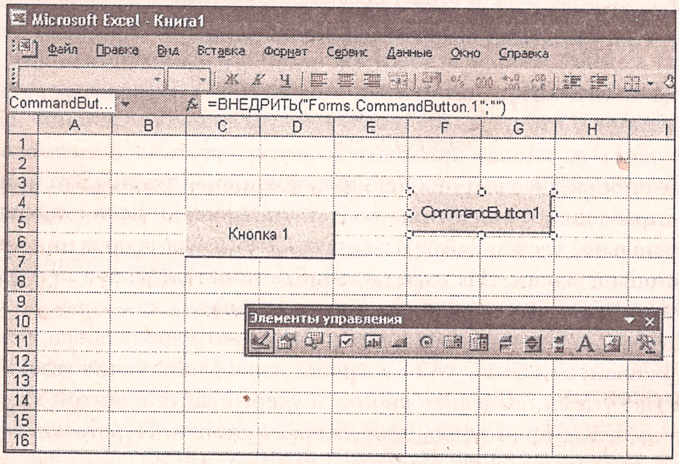


Рис. 1.13. Создание командной кнопки

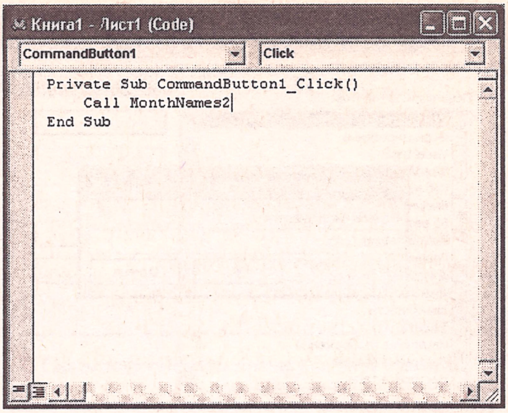


Рис. 1.14. Код обработки события для командной кнопки

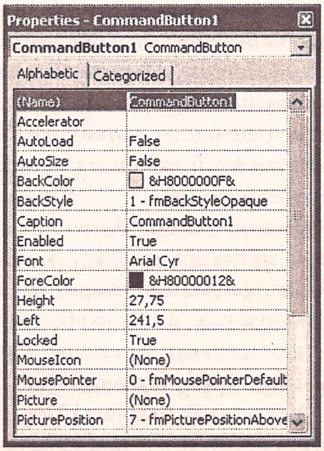


Рис. 1.15. Свойства объекта CommandButton

Для изменения текста на командной кнопке измените значение свойства `Caption`. Кроме этого, можно указать шрифт надписи и цвет фона и текста. Если кнопка должна удовлетворительно работать и в Excel 97, желательно изменить значение свойства `TakeFocusOnClick` на `False` (по умолчанию принято значение `True`). Если при щелчке кнопка получает фокус, Excel 97 не позволит присваивать значения некоторым свойствам, например, свойству `NumberFormat` объекта `Range`.

Панели инструментов

Если макрос необходимо связать с кнопкой на панели инструментов, можно модифицировать одну из встроенных панелей инструментов или создать собственную панель. Для создания собственной панели инструментов воспользуйтесь командой Вид⇨Панели инструментов⇨Настройка (View⇨Toolbars⇨Customize). Откроется окно Настройка (Customize). Щелкните на кнопке Создать (New) и введите имя новой панели инструментов (рис. 1.16).

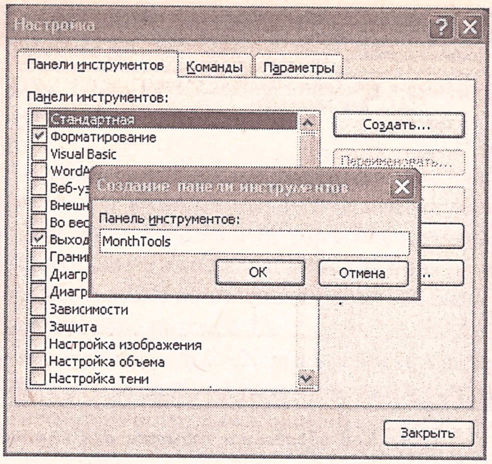


Рис. 1.16. Создание новой панели инструментов

В диалоговом окне Настройка (Customize) активизируйте вкладку Команды (Commands) и выберите категорию Макросы (Macros). Перетащите объект Настраиваемая кнопка (Custom Button) с пиктограммой улыбающегося лица на новую или уже существующую панель инструментов. После этого щелкните на кнопке Изменить выделенное (Modify-Selection) или щелкните правой кнопкой мыши на кнопке новой панели инструментов для получения доступа к контекстному меню.

Выберите пункт Назначить макрос (Assign Macro) и имя макроса (в данном случае это `MonthNames2`), который будет назначен кнопке. Кроме этого, щелкнув на кнопке Выбрать значок для кнопки (Change Button Image), можно изменить пиктограмму на кнопке. Можно назначить существующее изображение из предоставленной библиотеки или отредактировать изображение, щелкнув на кнопке Изменить значок на кнопке (Edit Button Image). Обратите внимание, что при внесении этих изменений диалоговое окно Настройка (Customize) должно оставаться открытым. Желательно ввести описательный текст в поле Имя (Name). Этот текст будет выводиться на экран во всплывающей подсказке. После этого диалоговое окно Настройка (Customize) можно закрыть.

Для запуска макроса выберите начальную ячейку для названий месяцев и щелкните на новой кнопке на панели инструментов. Если новую панель инструментов необходимо передать другим пользователям вместе с книгой, панель инструментов можно связать с книгой. После этого панель инструментов будет появляться на компьютере другого пользователя сразу при открытии книги (если панель инструментов с таким же именем еще не существует).

При связывании панели инструментов с книгой возникает потенциальная проблема. Так как Excel не заменяет существующие панели инструментов, может использоваться панель инструментов, присоединенная к более старой версии книги. Решение этой проблемы предлагается в следующем разделе при описании процедур обработки событий.

Для присоединения панели инструментов к активной книге воспользуйтесь командой Вид⇨Панели инструментов⇨Настройка (View⇨Toolbars⇨Customize). Откроется диалоговое окно Настройка (Customize). Активизируйте вкладку Панели инструментов (Toolbars) и щелкните на кнопке Вложить (Attach). Откроется диалоговое окно Управление панелями инструментов (Attach Toolbars), показанное на рис. 1.17.

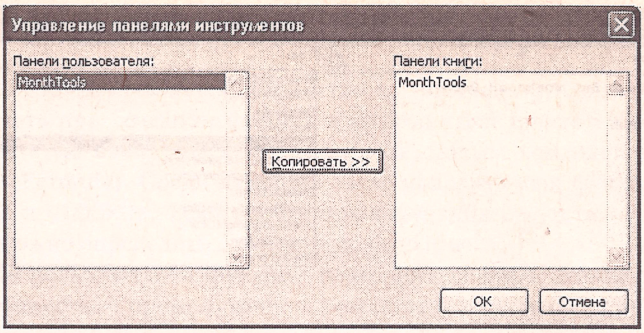


Рис. 1.17. Связывание панели инструментов с книгой

Выберите имя панели инструментов в левом списке и щелкните на кнопке Копировать >> (Copy >>).

Если к книге присоединена более старая версия панели инструментов, выберите панель инструментов в правом списке и щелкните на кнопке Удалить (Delete) (эта кнопка доступна при выборе пункта из правого списка) для ее удаления. После этого выберите имя панели инструментов в левом списке и еще раз щелкните на кнопке Копировать >> (Copy >>). Щелкните на кнопке OK для завершения подключения панели инструментов и выхода из диалогового окна Настройка (Customize).

Процедуры обработки событий

Процедуры обработки событий являются специальными функциями, которые выполняются в ответ на возникающие в Excel события. Среди событий можно выделить действия пользователя, например щелчок на кнопке, и действия системы, например пересчет содержимого листа. Начиная с Excel 97 в Excel предоставляется множество событий, для которых можно создавать обрабатывающий код.

Хорошим примером является процедура обработки события щелчка для командной кнопки ActiveX, запускающей макрос `MonthNames2`. Код для этой процедуры обработки события вводился в модуль кода, связанный с листом, в которой встроена командная кнопка. Все процедуры обработки событий хранятся в модуле кода, связанном с книгой, листом, диаграммой или диалоговым окном `UserForm`.

Для просмотра доступных событий можно активизировать модуль (например модуль `ЭтаКнига`), выбрать из левого раскрывающегося списка объект, например `Workbook`, и активизировать правый раскрывающийся список (рис. 1.18).

Процедура обработки события `Workbook_Open()` может использоваться для инициализации книги при открытии. Код может быть достаточно простым, например активизировать определенный лист или диапазон для ввода данных. Код может быть более сложным и обеспечивать создание новой панели меню для книги.

> Для сохранения совместимости с Excel 5 и Excel 95 в стандартном модуле можно создать подпрограмму, которая называется `Auto_Open()` и запускается при открытии книги. Если при этом одновременно существует процедура `Workbook_Open()`, она запускается в первую очередь.

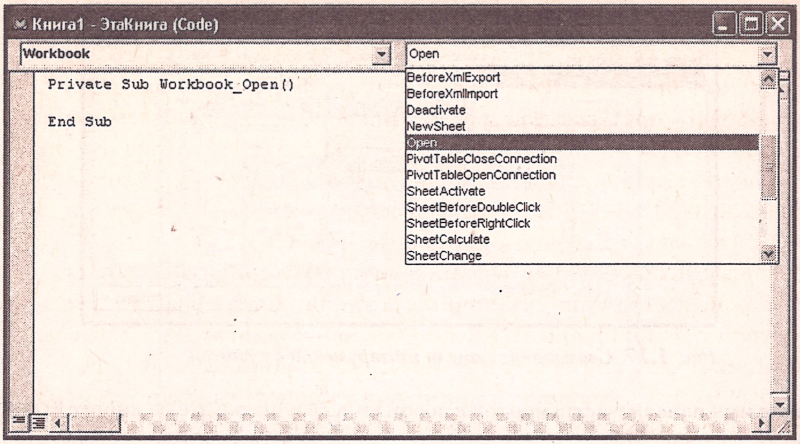


Рис. 1.18. События, доступные в модуле ЭтаКнига

Не сложно заметить, что выбор событий достаточно велик. Некоторые события, например BeforeSave и BeforeClose, допускают отмену события. Следующая процедура обработки события запрещает сохранение книги, пока в ячейке A1 листа Sheet1 не будет храниться значение True.

```
Private Sub Workbook_BeforeClose(Cancel As Boolean)
    If ThisWorkbook.Sheets("Sheet1").Range("A1").Value <> True _
        Then Cancel = True
End Sub
```

Кроме этого, такой код предотвращает закрытие окна Excel.

Удаление присоединенной панели инструментов

Как было показано ранее в этой главе, если к книге присоединена панель инструментов, при отправке новой версии панели инструментов вместе с книгой может возникнуть проблема. Такая же проблема возникает, если пользователь сохраняет книгу под другим именем. При открытии новой книги существующая панель инструментов не замещается, а старая панель инструментов запускает макросы из старой книги.

Одним из решений, позволяющих снизить вероятность этой проблемы, является удаление собственной панели инструментов при закрытии книги:

```
Private Sub Workbook_BeforeClose (Cancel As Boolean)
    On Error Resume Next
    Application.CommandBars("MonthTools").Delete
End Sub
```

Оператор On Error обрабатывает ситуацию, когда пользователь вручную удаляет панель инструментов перед закрытием книги. Если не добавить оператор On Error, то при попытке удаления отсутствующей панели инструментов приложение выдаст сообщение об ошибке времени выполнения. Оператор On Error Resume Next заставляет приложение игнорировать сообщения об ошибках и продолжать выполнение со следующей строки кода.

Определенные пользователем функции

В составе Excel предоставляется множество встроенных функций листов, которые можно использовать при создании формул в ячейках. Для просмотра списка функций можно выбрать пустую ячейку на листе и воспользоваться командой меню Вставка⇨Формула (Insert⇨Formula). Среди наиболее часто используемых функций можно выделить СУММ, ЕСЛИ и VLOOKUP. Если необходимая функция отсутствует в составе Excel, с помощью VBA можно определить собственную функцию.

Определенные пользователем функции позволяют снизить сложность листа. Сложные расчеты с множеством промежуточных результатов в нескольких ячейках можно упростить до единственной формулы в одной ячейке. Кроме этого, определенные пользователем функции можно применять для повышения производительности пользователей, если некоторые из них применяют одни и те же процедуры расчетов. Можно даже создать библиотеку функций, специфичных для конкретной организации.

Создание определенных пользователем функций

В отличие от ручных операций, определенные пользователем функции записать невозможно. Такие функции необходимо создавать с нуля на основе стандартного модуля в редакторе VBE. Для вставки стандартного модуля необходимо щелкнуть правой кнопкой мыши на окне Project Explorer (Окне проекта) и выбрать пункт Insert⇨Module (Вставка⇨Модуль). Простой пример определенной пользователем функции показан ниже:

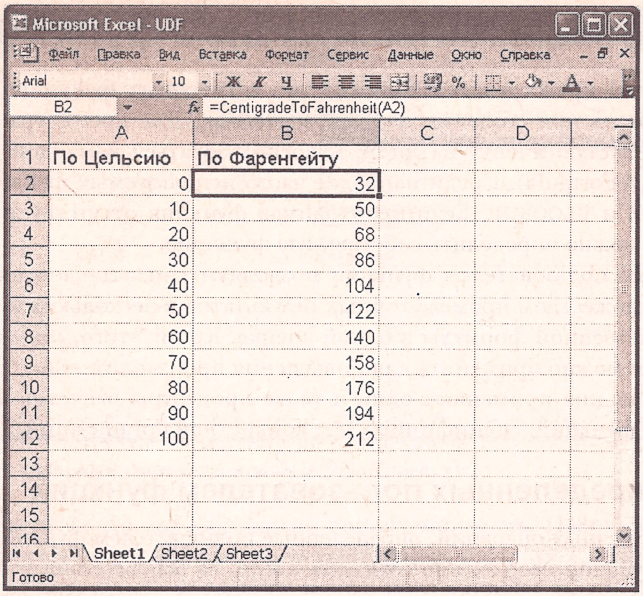
```
Function CentigradeToFahrenheit (Centigrade)
   CentigradeToFahrenheit = Centigrade * 9 / 5 + 32
End Function
```

Функция CentigradeToFahrenheit() предназначена для преобразования градусов по Цельсию в градусы по Фаренгейту. В столбце А на листе можно указать значения температуры по шкале Цельсия и воспользоваться функцией CentigradeToFahrenheit() для получения соответствующих значений температуры по шкале Фаренгейта в столбце В. Если выделить ячейку В2, то определенная пользователем формула будет показана в строке Формула (Formula) (рис. 1.19).

В ячейки диапазона В3:В13 была скопирована формула.

Главным отличием подпрограммы от функции является возвращаемое значение функции. Функция CentigradeToFahrenheit() возвращает числовое значение, выводимое в ячейке листа, в которой указана функция CentigradeToFahrenheit(). Для возврата значения оно присваивается имени функции.

Функции обычно получают один или несколько входных параметров. Функция CentigradeToFahrenheit() получает один параметр, который называется Centigrade. Этот параметр используется для расчета возвращаемого значения. При вводе формулы =CentigradeToFahrenheit(A2) значение ячейки А2 передается функции в качестве значения параметра Centigrade. Если в качестве параметра функции передается значение 0, функция возвращает значение 32. Результат передается обратно и выдается в виде значения ячейки В2, как показано ранее. То же происходит в каждой ячейке, содержащей ссылку на формулу =CentigradeToFahrenheit().



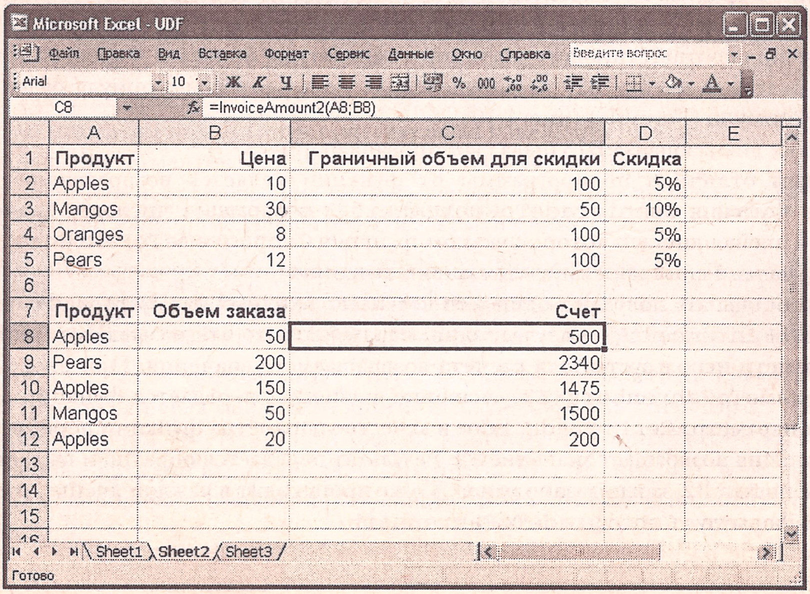
Microsoft Excel - UDF

B2 =CentigradeToFahrenheit(A2)

	A	B
1	По Цельсию	По Фаренгейту
2	0	32
3	10	50
4	20	68
5	30	86
6	40	104
7	50	122
8	60	140
9	70	158
10	80	176
11	90	194
12	100	212

Рис. 1.19. Использование определенной пользователем формулы

Еще один пример снижения сложности формул на листе показан на рис. 1.20. Таблица поиска в ячейках A2:D5 содержит цены каждого продукта, граничный объем для получения скидки и процент скидки после превышения граничного объема. При использовании обычных формул листа для расчета суммы пользователям пришлось бы применять комбинацию из трех формул поиска с логическими проверками.



Microsoft Excel - UDF

C8 =InvoiceAmount2(A8;B8)

	A	B	C	D
1	Продукт	Цена	Граничный объем для скидки	Скидка
2	Apples	10	100	5%
3	Mangos	30	50	10%
4	Oranges	8	100	5%
5	Pears	12	100	5%
6				
7	Продукт	Объем заказа	Счет	
8	Apples	50	500	
9	Pears	200	2340	
10	Apples	150	1475	
11	Mangos	50	1500	
12	Apples	20	200	

Рис. 1.20. Использование формул для снижения сложности расчетов

Функция `InvoiceAmount()` использует три параметра: `Product` содержит имя продукта, `Volume` содержит количество продаваемых единиц, `Table` содержит таблицу поиска. Формула в ячейке C8 определяет диапазоны для каждого параметра:

```
Function InvoiceAmount(ByVal Product As String, _
    ByVal Volume As Integer, ByVal Table As Range)

  ' Поиск цены в таблице
    Price = WorksheetFunction.VLookup(Product, Table, 2)

  ' Поиск граничного объема для получения скидки
    DiscountVolume = WorksheetFunction.VLookup(Product, Table, 3)

  ' Сравнение объема с граничным объемом для определения
  ' возможности предоставить скидку
    If Volume > DiscountVolume Then
    ' подсчитать цену со скидкой
        DiscountPercent = WorksheetFunction.VLookup(Product, Table, 4)
        InvoiceAmount = Price * DiscountVolume + Price * _
            (1 - DiscountPercent) * (Volume - DiscountVolume)
    Else
    ' подсчитать цену без скидки
        InvoiceAmount = Price * Volume
    End If
End Function
```

Используется абсолютный диапазон таблицы, что позволяет формулам ниже ячейки C8 находиться в одном и том же диапазоне. Сначала в функции используется функция `VLookup`, которая позволяет найти продукт в таблице поиска и получить соответствующее значение из второго столбца таблицы. Полученное значение присваивается переменной `Price`.

Если в процедуре VBA необходимо использовать функции листа Excel, интерпретатор VBA должен знать, где найти эту функцию. Для этого перед именем функции необходимо добавить `WorksheetFunction` *и точку. Для сохранения совместимости с Excel 5 и Excel 95 вместо* `WorksheetFunction` *можно указать* `Application`. *Не все функции листа доступны таким образом. Если функция оказывается недоступной, эквивалентная ей функция (или математический оператор) доступна в VBA и выполняет ту же операцию.*

В следующей строке функции выполняется поиск граничного объема для получения скидки. Полученное значение граничного объема присваивается переменной `DiscountVolume`. Проверка `If` в следующей строке сравнивает продаваемый объем в переменной `Volume` со значением переменной `DiscountVolume`. Если значение `Volume` больше, чем `DiscountVolume`, выполняются расчеты до оператора `Else`. В противном случае, выполняются расчеты после оператора `Else`.

Если значение `Volume` оказывается больше `DiscountVolume`, в таблице выполняется поиск процента скидки и найденное значение присваивается переменной `DiscountPercent`. После этого рассчитывается стоимость заказа. Для этого объем заказа в переменной `DiscountVolume` умножается на цену единицы товара. После этого к стоимости заказа добавляется цена со скидкой всех единиц товара свыше `DiscountVolume`. Обратите внимание на использование символа подчеркивания, перед которым указывается символ пробела. Такая комбинация обозначает продолжение кода на следующей строке.

Результат присваивается переменной с именем функции, InvoiceAmount. Значение этой переменной будет возвращено в ячейку на листе. Если значение переменной Volume не превышает значение переменной DiscountVolume, стоимость заказа рассчитывается как произведение цены и количества единиц продукта. Значение произведения присваивается переменной с именем функции.

Непосредственная ссылка на диапазоны

При определении функции вместо использования параметров можно непосредственно ссылаться на диапазоны листа. Эта возможность демонстрируется в следующей функции InvoiceAmount2():

```
Public Function InvoiceAmount2(ByVal Product As String, _
    ByVal Volume As Integer) As Double
    Dim Table As Range
    Set Table = ThisWorkbook.Worksheets("Sheet2").Range("A2:D5")
    Dim Price As Integer
    Price = WorksheetFunction.VLookup(Product, Table, 2)

    Dim DiscountVolume As Integer
    DiscountVolume = WorksheetFunction.VLookup(Product, Table, 3)

    Dim DiscountPercent As Double
    If Volume > DiscountVolume Then
      DiscountPercent = WorksheetFunction.VLookup(Product, Table, 4)
      InvoiceAmount2 = Price * DiscountVolume + Price * _
        (1 - DiscountPercent) * (Volume - DiscountVolume)
    Else
        InvoiceAmount2 = Price * Volume
    End If
End Function
```

Обратите внимание, что переменная Table больше не является параметром функции. Вместо этого оператор Set определяет переменную Table с помощью непосредственной ссылки на диапазон листа. Хотя этот метод работоспособен, возвращаемое значение функции не будет пересчитываться при модификации содержимого таблицы поиска. Excel не осознает необходимости пересчета функции при изменении содержимого таблицы, так как не знает, что таблица используется в функции.

Excel пересчитывает определенные пользователем функции только при изменении входных параметров. Если таблицу поиска необходимо исключить из списка параметров функции, но при этом сохранить возможность автоматического пересчета, функцию можно определить, как volatile. Такое определение функции показано ниже:

```
Public Function InvoiceAmount2(ByVal Prod As String,  _
       ByVal Volume As Integer) As Double

Application.Volatile

    Dim Table As Range
    Set Table=ThisWorkbook.Worksheets("Sheet2").Range("A2:D5")
...
```

Но стоит обратить внимание, что такая возможность имеет свою цену. Если определенная пользователем функции объявлена, как volatile, функция будет пересчитываться при изменении любого значения на листе. Если функция используется в большом количестве ячеек, накладные расходы на пересчет функций могут стать заметны.

Чего не могут определенные пользователем функции

Распространенной ошибкой многих пользователей является попытка создания функции, которая меняет структуру листа (например, путем копирования диапазона ячеек). Такие попытки завершаются неудачей. Сообщения об ошибках не выдаются, так как Excel просто игнорирует соответствующие строки кода и причина неудачи оказывается неочевидна.

> Вызываемые из ячеек листа определенные пользователем функции не могут менять структуру листа. Это значит, что такая функция не может вернуть значение не в ту ячейку, из которой она была вызвана. Кроме этого, такая функция не изменяет физические характеристики ячейки, например цвет шрифта или цвет фона. Также определенные пользователем функции не выполняют такие операции, как копирование и перемещение ячеек листа. Определенные пользователем функции даже не могут выполнять операции, неявно подразумевающие перемещение курсора, например, вызывать команду Правка⇨Найти (Edit⇨Find). Определенная пользователем функция может вызывать другую функцию или даже подпрограмму, но эти процедуры оказываются связаны теми же ограничениями, что и определенные пользователем функции. Им также нельзя модифицировать структуру листа.

Интерпретатор VBA в Excel различает определенные пользователем функции, которые применяются в ячейках листа, и функции, не связанные с ячейками листа. Если исходная процедура не являлась определенной пользователем функцией в ячейке листа, функция может выполнять все допустимые в Excel операции, как и обычная подпрограмма.

Стоит обратить внимание, что определенные пользователем функции работают не так эффективно, как встроенные функции листа Excel. Если на листе применяется большое количество определенных пользователем функций, время пересчета листа окажется больше, чем при использовании такого же количества встроенных функций Excel.

Объектная модель Excel

Язык программирования Visual Basic for Application используется во всех приложениях из состава Microsoft Office. Кроме Excel, язык VBA может применяться в Word, Access, PowerPoint, Outlook, FrontPage и Project. Изучив язык, его можно использовать в любом из этих приложений. Но для работы с конкретным приложением необходимо ознакомиться со списком доступных объектов. В Word приходится работать с документами, абзацами и словами. В Access предоставляются объекты баз данных, наборов записей и полей. В Excel доступны книги, листы и диапазоны.

В отличие от множества других языков программирования, в VBA для Office нет необходимости создавать собственные объекты. В приложении четко определено множество объектов, организованных в определенную иерархию. Эта структура называется объектной моделью приложения. Этот раздел можно рассматривать как введение в объектную модель Excel. Полное описание объектной модели приводится в приложении А (это приложение также доступно на сайте `www.wrox.com`).

Объекты

Для начала рассмотрим базовые понятия, характерные для объектно-ориентированного программирования. Это не формальное описание, и его достаточно для понимания принципов работы с объектами Excel.

В основе объектно-ориентированного программирования лежит возможность деления всех известных сущностей на классы. Экземпляры классов называются объектами. Вы и я являемся объектами класса Личность. Такими же объектами являются мир и все-

ленная. В Excel объектами являются книга, лист и диапазон. Кроме перечисленных объектов, в объектной модели Excel доступно еще около 200 объектов. Рассмотрим ряд примеров использования объекта Range в коде VBA. Ссылка на ячейки диапазона B2:C4 может выглядеть следующим образом:

```
Range("B2:C4")
```

Если диапазону ячеек присвоить имя Data, это имя можно использовать таким же образом:

```
Range("Data")
```

Кроме этого, существуют способы ссылаться на текущую активную ячейку и текущее выделение.

В ситуации, показанной на рис. 1.21, свойство ActiveCell ссылается на ячейку B2, а свойство Selection — на диапазон B2:E6.

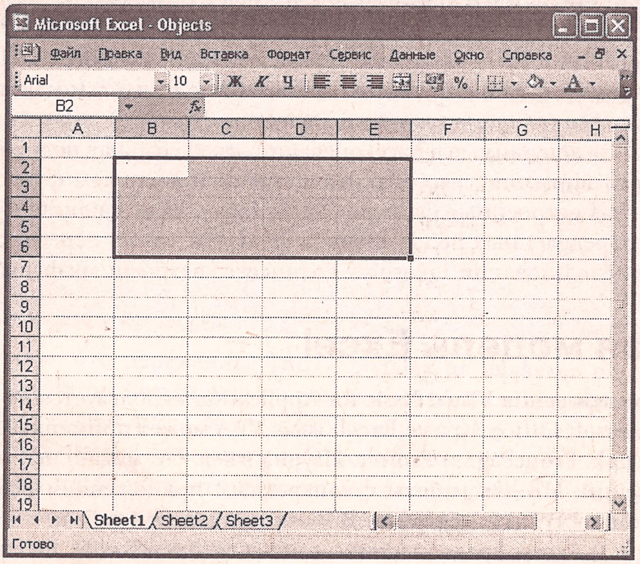


Рис. 1.21. Разница между текущим выделением и активной ячейкой

Коллекции

В коллекцию входит несколько объектов. Городской квартал является коллекцией зданий. Здание является коллекцией объектов этажей. Коллекция объектов сама является объектом, хранящим другие, тесно связанные, объекты. Коллекции и объекты часто формируют иерархическую структуру или структуру в виде дерева.

Даже приложение Excel является объектом. Этот объект называется Application. В него входит коллекция Workbooks, в которой хранятся объекты Workbook для всех открытых книг. Каждый объект Workbook содержит коллекцию Worksheets, в которой хранятся объекты Worksheet для листов этой книги.

> Обратите внимание на множественное число в названии объекта Worksheets (коллекция) и единственное число в названии объекта Worksheet (объект). На самом деле это разные объекты.

Если необходимо сослаться на члена коллекции, можно указать его позицию в коллекции (номер индекса, начиная с 1) или имя (текст в кавычках). Если в определенный момент открыта только одна книга, которая называется `Data.xls`, на нее можно сослаться с использованием следующих операторов:

```
Workbooks(1)
Workbooks("Data.xls")
```

Если в активной книге открыто четыре листа, называемые `North`, `South`, `East` и `West` (именно в таком порядке), на второй лист можно сослаться одним из следующих операторов:

```
Worksheets(2)
Worksheets("South")
```

Если необходимо сослаться на лист `DataInput` в книге `Sales.xls`, но книга `Sales.xls` не является активной, ссылку на лист необходимо квалифицировать с помощью ссылки на книгу, разделив квалификаторы точкой, как показано ниже:

```
Workbooks("Sales.xls").Worksheets("DataInput")
```

С точки зрения объектно-ориентированного программирования символы `>`, `<`, `=` и `.` называются операторами. Точка (`.`) называется оператором членства. Например, можно сказать, что коллекция `Worksheets` является членом класса `Workbook`.

Если в определенный момент активна другая книга, то для ссылки на ячейку B2 на листе `DataInput` необходимо использовать следующий код:

```
Workbooks("Sales.xls").Worksheets("DataInput").Range("B2")
```

Предполагается, что оператор `Workbooks("Sales.xls")` возвращает экземпляр класса `Workbook`. Оператор `.Worksheets("DataInput")` возвращает единственный объект `Worksheet`, а оператор `.Range("B2")` — единственный диапазон. Интуитивно можно догадаться, что вся строка извлекает один объект `Workbook`, один объект `Worksheet` и конкретный диапазон.

Теперь можно более подробно рассмотреть объекты и методы манипуляции объектами из кода VBA. Существует четыре ключевых характеристики объекта, о которых нужно помнить. Это поля, свойства, методы и события, связанные с объектом.

Поля

Поля представляют собой переменные, хранящие состояние объекта. Поля могут содержать любую информацию, которую должны знать экземпляры классов. Например, класс `Customer` должен знать имя потребителя. Выражение "должен знать" используется из-за того, что необходимая информация зависит от проблемной области. Если расширить данный пример, то класс `Customer` должен получать доступ к информации Employer Identification Number (EIN). Но если потребители не являются юридическими лицами, номер EIN не имеет никакого значения, а номер социального страхования имел бы больший смысл в данном случае.

Вот пример поля EIN:

```
Private FEmploymentIdentificationNumber As String
```

Следуя этому соглашению, будет использован префикс `F-` для того, чтобы указывать, что имя является именем поля (для свойств префикс `F-` не используется; свойства рассматриваются в следующем разделе). Еще одно соглашение предполагает, что поля объявляются частными и доступ к ним осуществляется опосредованно через свойства. Модификаторы доступа (например, `Private`) более подробно рассматриваются в главе 5.

Свойства

Свойства являются результатом определенного развития. Изначально классы имели только поля и методы. Через некоторое время оказалось, что для обеспечения допустимости присваиваемых значений поля нуждались в соответствующих методах. Свойства стали результатом смешения простоты использования и проверки действительности. Свойства на самом деле являются специальными методами, которые для потребителя (пользователя класса) выглядят как поля, но ведут себя и программируются создателем (автором класса) как методы. Свойства обеспечивают управление доступом к информации о состоянии объекта.

Следовательно, свойства являются атрибутами, как и поля. В свойствах неявно комбинируется поведение вызова метода и простота обращения к полю. Существует соглашение, по которому имена свойств совпадают с именами полей без префикса `F-`. Другие авторы книги могут придерживаться других соглашений. Вместо того чтобы делать вид, что различия в соглашениях по именованию не существуют, можно выбрать подходящее соглашение и следовать ему при создании собственного кода. (В других соглашениях по именованию префиксы используются для предоставления информации о типе данных поля или свойства. Выберите одно из соглашений и применяйте его постоянно.)

Свойства в Excel можно рассмотреть на примере объекта `Range`. Объект `Range` имеет свойство `RowHeight` и свойство `ColumnWidth`. Объект `Workbook` имеет свойство `Name`, в котором хранится имя файла. Некоторые свойства предоставляют механизм изменения значения, например, свойство `ColumnWidth` объекта `Range`. Для этого свойству можно просто присвоить новое значение. Другие свойства, например, свойство `Name` объекта `Workbook`, предназначены только для чтения. Для изменения значения свойства `Name` недостаточно просто присвоить новое значение.

Ссылка на свойство состоит из имени объекта, после которого указывается название свойства. Имена объекта и свойства разделяются точкой (оператором членства). Например, для изменения ширины столбца, в который входит активная ячейка, и установки ширины в 20 пунктов свойству `ColumnWidth` объекта `ActiveCell` необходимо присвоить значение 20:

```
ActiveCell.ColumnWidth = 20
```

Для ввода имени "Florence" в ячейку C10 это имя присваивается свойству `Value` объекта `Range`:

```
Range("C10").Value = "Florence"
```

Если интересующий объект `Range` находится не на активном листе активной книги, необходима более конкретная ссылка на объект:

```
Workbooks("Sales.xls").Worksheets("DataInput").Range("C10").Value = 10
```

> Интерпретатор VBA позволяет выполнять операции, которые невозможно выполнить вручную. Интерпретатор позволяет вводить данные в листы, не отображаемые на экране. Копирование и перемещение данных может происходить без переключения активного листа. Таким образом, для манипулирования данными с помощью VBA активизация книги, листа или диапазона требуется очень редко. Чем меньше приходится активизировать объекты, тем быстрее будет работать код. К сожалению, механизм записи макросов записывает только действия пользователя и постоянно использует механизмы активизации объектов.

В предыдущих примерах было показано, как присваивать значения свойствам объектов. Кроме этого, значения свойств объектов можно присваивать переменным или свойствам других объектов. Следующий код можно использовать для непосредственного присвоения ширины одной ячейки на активном листе ширине другой ячейки:

```
Range("C1").ColumnWidth = Range("A1").ColumnWidth
```

Значение ячейки C1 активного листа можно присвоить ячейке D10 на листе, который называется `Sales`. Для этого можно применить следующий код:

```
Worksheets("Sales").Range("D10").Value = Range("C1").Value
```

Значение свойства можно присвоить переменной для дальнейшего использования в коде. В этом примере текущее значение ячейки M100 сохраняется в переменной, ячейке M100 присваивается новое значение, на экран выводится автоматически пересчитанный результат, и ячейке M100 присваивается первоначальное значение:

```
OpeningStock = Range("M100").Value
Range("M100").Value = 100
ActiveSheet.PrintOut
Range("M100").Value = OpeningStock
```

Некоторые свойства предназначены только для чтения. Это значит, что непосредственное присвоение значения этим свойствам невозможно. Иногда существует опосредованный способ. В качестве примера можно привести свойство `Text` объекта `Range`. Для ввода значения в ячейку оно присваивается свойству `Value`. Для определения формата отображения чисел в ячейке используется свойство `NumberFormat`. Свойство `Text` позволяет получить уже отформатированное содержимое ячейки. В следующем примере в окне сообщения выводится строка $12,345.60:

```
Range("B10").Value = 12345.6
Range("B10").NumberFormat = "$#.##0.00"
MsgBox Range("B10").Text
```

Это единственный способ, с помощью которого можно изменить значение свойства `Text`.

Методы

В то время как свойства являются квалифицирующими характеристиками объектов, методы являются операциями, которые могут выполняться объектами или над объектами. С лингвистической точки зрения классы можно рассматривать как существительные, объекты — как экземпляры существительных, поля и свойства — как прилагательные, а методы — как глаголы. Методы часто используются для изменения свойств объекта. Зоолог мог бы определить класс Человек разумный с глаголом Ходить. Глагол Ходить мог бы реализовываться в терминах Скорости, Расстояния и Направления, которые позволяют получить новое Положение. Компания по выпуску кредитных карт могла бы реализовать класс Клиент с методом Расход. Оплата за товары (метод Расход) может приводить к уменьшению доступной кредитной линии (свойство ДоступныйКредит).

Простым примером метода в Excel является метод `Select` объекта `Range`. На метод можно ссылаться как и на свойство: сначала указывается имя объекта, а потом точка и имя метода (а также необходимые параметры). Возвращаясь к методу Ходить, можно выделить такие параметры, как расстояние, направление и время работы метода. Результатом работы метода будет новое Положение (предполагается, что известно текущее положение). Следующий код позволяет выделить ячейку G4:

```
Range("G4").Select
```

Еще одним примером является метод Copy объекта Range. Следующий код копирует диапазон A1:B3 в буфер обмена:

```
Range("A1:B3").Copy
```

Часто методы получают параметры (или аргументы), используемые для управления работой метода. Например, метод Paste объекта Worksheet позволяет вставлять содержимое буфера обмена в лист. Но если не указать место, в которое будут вставляться данные, верхний левый угол вставляемых данных будет находиться в активной ячейке. Для переопределения такого поведения используется параметр Destination (параметры рассматриваются далее в этом разделе):

```
ActiveSheet.Paste Destination:=Range("G4")
```

> Обратите внимание, что значения параметров указываются с помощью оператора :=, а не =.

Часто методы Excel позволяют применять короткую форму записи. Предыдущие примеры использования методов Copy и Paste могут быть записаны одной строкой:

```
Range("A1:B3").Copy Destination:=Range("G4")
```

Этот код оказывается значительно эффективнее кода, который создается механизмом записи макросов:

```
Range("A1:B3").Select
Selection.Copy
Range("G4").Select
ActiveSheet.Paste
```

События

Еще одной важной концепцией языка VBA являются события, на которые могут реагировать объекты. Щелчок мышью на командной кнопке, двойной щелчок на ячейке, пересчет листа и открытие или закрытие книги являются примерами событий. На простом языке события являются тем, что происходит. В контексте языка VBA события являются тем, что происходит с объектами.

Все элементы управления ActiveX, доступные в панели инструментов Элементы управления (Control Toolbox), могут реагировать на события. Эти элементы управления могут использоваться как на листах, так и в диалоговых окнах UserForm для расширения функциональности этих объектов.

Листы и книги также реагируют на целый ряд событий. Если объект должен реагировать на определенное событие, для этого события необходимо написать процедуру обработки. На самом деле события являются не чем иным, как адресами (числами). Все сущности в компьютере являются последовательностями чисел. Для компьютера главное, как используются эти числа. Номер события является адресом (тоже числом), который ссылается на адрес (еще одно число) метода. При возникновении события связанный с ним адрес используется для вызова метода, на который ссылается этот адрес. Эти методы называются обработчиками событий. К счастью, интерпретатор самостоятельно справляется с выделением и назначением этих адресов. От нас требуется только знание синтаксиса, используемого, чтобы сообщить объекту, что определенный обработчик должен вызываться при возникновении соответствующего события. К счастью, среда VBE делает еще один шаг навстречу: редактор VBE автоматически генерирует и назначает обработчики событий. Конечно, программист может назначать обработчики вручную, что оказывается полезным при работе с классами, которые не входят в библиотеки Excel или ActiveX.

Например, можно определить, что пользователь выделил новую ячейку и подсветить весь столбец или строку, в которой находится эта ячейка. Для этого редактор VBE должен сгенерировать обработчик события SelectionChange, а программист — написать код обработчика события. Существующее соглашение по именованию предполагает, что редактор VBE указывает имя объекта, символ подчёркивания и имя события. Таким образом, обработчик события SelectionChange будет называться Worksheet_SelectionChange. Выполнение показанной ниже последовательности действий приведет к создания обработчика события:

- сначала активизируйте окно редактора VBE, нажав комбинацию клавиш <Alt+F11>;
- в левом раскрывающемся списке Object (Объект) выберите пункт Worksheet, а в правом раскрывающемся списке Procedure (Процедура) выберите пункт SelectionChange;
- в сгенерированной подпрограмме введите следующий код:

```
Private Sub Worksheet_SelectionChange(ByVal Target As Range)
    Rows.Interior.ColorIndex = xlColorIndexNone
    Target.EntireColumn.Interior.ColorIndex = 36
    Target.EntireRow.Interior.ColorIndex = 36
End Sub
```

Этот обработчик события выполняется каждый раз, когда пользователь выделяет новую ячейку или блок ячеек. Параметр Target ссылается на выделенный диапазон, представленный объектом Range. Первый оператор сбрасывает свойство ColorIndex всех ячеек листа (ячейки теряют цвет фона). Второй и третий операторы устанавливают цвет фона всей строки и всего столбца, в которых находятся выделенные ячейки (цвет фона делается бледно-желтым). Если палитра цветов Excel была модифицирована, предопределенная константа xlColorIndexNone может не соответствовать кремовому цвету.

В этом примере свойства используются более сложным образом, чем было показано ранее. Рассмотрим отдельные составляющие. Если предположить, что параметр Target является объектом Range, который соответствует ячейке B10, то следующий код использует свойство EntireColumn объекта Range ячейки B10 для получения ссылки на весь столбец B. Столбцу B соответствует диапазон B1:B65536 или, в коротком варианте, B:B.

```
Target.EntireColumn.Interior.ColorIndex = 36
```

Точно так же следующая строка кода меняет цвет строки 10, которой соответствует диапазон A10:IV10 или, в коротком варианте; 10:10.

```
Target.EntireRow.Interior.ColorIndex = 36
```

Свойство Interior объекта Range ссылается на объект Interior, описывающий фон диапазона. Наконец, свойство ColorIndex объекта Interior устанавливается равным индексу интересующего цвета.

Многим этот код может показаться неинтуитивным, поэтому возникает вопрос, как получить информацию об использовании определенного объекта Excel?

Получение справки

Простым способом получения кода, соответствующего определенной операции, является использование механизма записи макросов. Записанный код скорее всего оказывается неэффективным, но в таком коде можно обнаружить ссылки на необходимые объекты и используемые свойства и методы. Если включить механизм записи макроса и изменить цвет фона ячейки, в результате будет получен следующий код:

```
With Selection.Interior
     .ColorIndex = 36
     .Pattern = xlSolid
End With
```

Конструкция `With...End With` рассматривается более подробно далее в этой главе. Этот код эквивалентен следующей конструкции:

```
Selection.Interior.ColorIndex = 36
Selection.Interior.Pattern = xlSolid
```

Вторую строку вводить необязательно, так как сплошная заливка используется по умолчанию. Механизм записи макроса явно указывает все значения и не использует неявные значения и поведение объектов. В первой строке можно обнаружить информацию, полезную для создания собственного кода. В данном случае необходимо узнать, как превратить объект `Range` (`Selection`) в целую строку или целый столбец. Если это возможно, для этого придется использовать свойство или метод объекта`Range`.

Окно Object Browser

Окно Object Browser (Просмотр объектов) является незаменимым инструментом для получения информации о полях, свойствах, методах и событиях объектов Excel. Для отображения окна Object Browser (Просмотр объектов) необходимо переключиться в окно редактора VBE. Для этого можно воспользоваться командой меню View⇨Object Browser (Вид⇨Просмотр объектов), нажать клавишу <F2> или щелкнуть на кнопке Object Browser (Просмотр объектов) на панели инструментов Standard (Стандартная). В результате откроется окно, показанное на рис. 1.22.

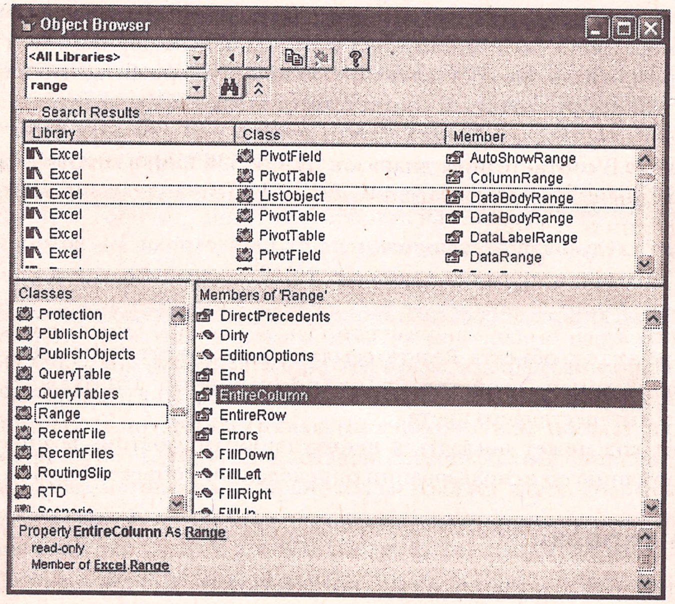


Рис. 1.22. Окно Object Browser

Объекты перечислены в окне, которое называется Classes (Классы). Для быстрого перехода к объекту `Range` можно щелкнуть на окне и нажать клавишу <r>.

С другой стороны, можно щелкнуть на поле поиска (возле пиктограммы в виде бинокля) и ввести **range**. После нажатия клавиши <Enter> или щелчка на пиктограмме с изображением бинокля будет показан список объектов, содержащих в имени введенный фрагмент. При щелчке на объекте Range в окне результатов поиска, в окне Classes (Классы) будет выделен объект Range. Этот прием оказывается полезным при поиске информации об определенном свойстве, методе или событии.

После этого становится доступен список всех полей, свойств, методов и событий этого объекта, отсортированный в алфавитном порядке. Щелчок на списке правой кнопкой мыши позволяет выбрать пункт Group Members (Члены групп) для разделения свойств, методов и событий. Такое разделение значительно упрощает чтение. Беглый просмотр списка позволяет найти свойства EntireColumn и EntireRow, которые выглядят реальными кандидатами для решения поставленной задачи. Для подтверждения выбора выберите свойство EntireColumn и щелкните на пиктограмме ? в верхней части окна Object Browser (Просмотр объектов). Откроется окно, показанное на рис. 1.23.

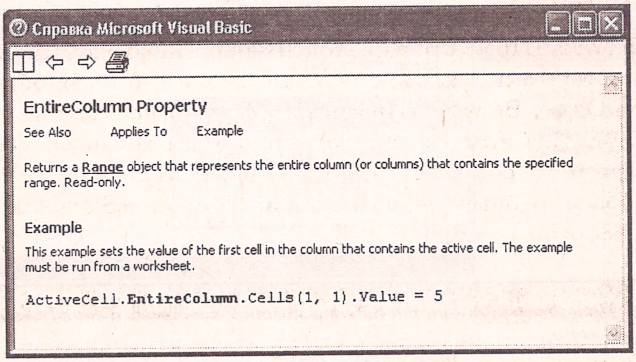


Рис. 1.23. Окно со справочной информацией

Часто этот метод можно использовать для доступа к информации о других объектах и методах. На данный момент достаточно связать найденные свойства и применить их вместе с подходящим объектом.

Эксперименты в окне Immediate

Если возникло желание поэкспериментировать с кодом, можно воспользоваться окном Immediate (Проверка) в редакторе VBE. Примените команду меню View⇨Immediate Window (Вид⇨Окно Проверка), нажмите комбинацию клавиш <Ctrl+G> или щелкните на кнопке Immediate Window (Окно Проверка) на панели инструментов Debug (Отладка). После этого окно Immediate (Проверка) появится на экране. Окно Excel и окно редактора VBE можно расположить на экране рядом, что позволит вводить команды в окне Immediate (Проверка) и видеть результат их выполнения в окне Excel (рис. 1.24).

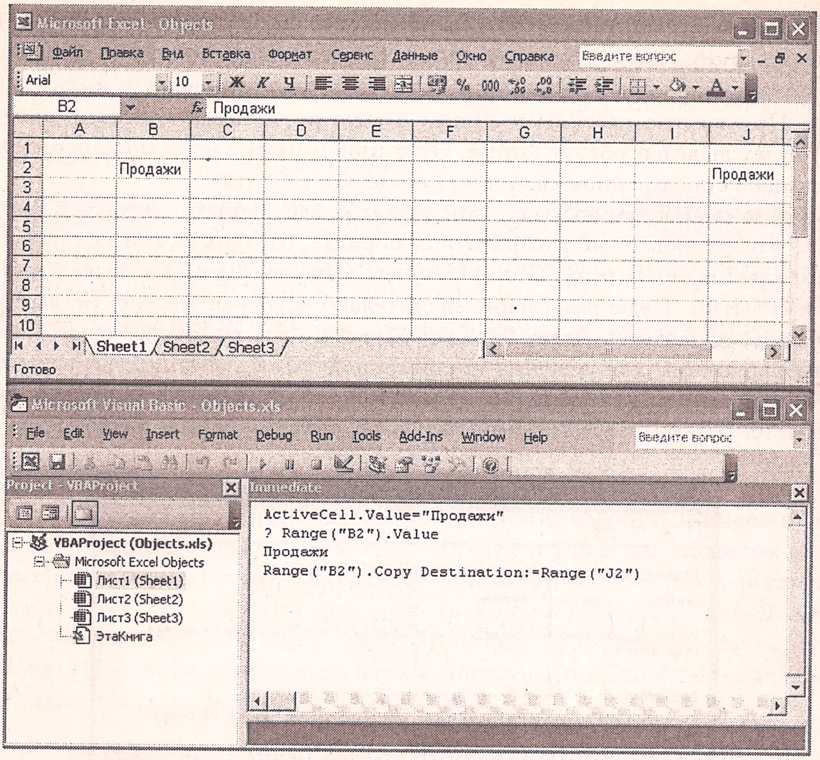


Рис. 1.24. Просмотр результата работы команд, вводимых в окне Immediate

При вводе команды в окне Immediate (Проверка) (нижний правый угол рис. 1.24) и нажатии клавиши <Enter> команда выполняется немедленно. Для повторного выполнения одной и той же команды щелкните на строке команды и еще раз нажмите клавишу <Enter>.

В данном случае свойству `Value` объекта `ActiveCell` присваивается значение `"Продажи"`. Если нужно вывести значение, перед кодом необходимо ввести знак вопроса, который является синонимом команды `Print`:

```
?Range("B2").Value ' Эквивалентно команде Print Range("B2").Value
```

Этот код распечатает слово "Продажи" в следующей строке окна Immediate (Проверка). Последняя команда скопирует значение из ячейки B2 в ячейку J2.

Язык VBA

В этом разделе будут показаны компоненты языка VBA, которые присутствуют во всех версиях Visual Basic и во всех приложениях Microsoft Office, а также использованы примеры на основе объектной модели Excel. Но главная цель — демонстрация общего синтаксиса. Многие структуры и концепции применяются и в других языках программирования, хотя синтаксис может отличаться. Здесь будут рассмотрены следующие вопросы:

- сохранение информации в переменных и массивах;
- условные операторы;
- использование циклов;
- базовые механизмы обработки ошибок.

Базовый ввод и вывод

Для начала рассмотрим простые способы коммуникации с пользователем, которые сделают макросы более гибкими и полезными. Если необходимо выдать сообщение, применяется функция MsgBox. Сообщение может использоваться для выдачи предупреждения или получения ответа на простой вопрос.

В первом примере перед началом операции печати необходимо убедиться в доступности принтера. С помощью следующего кода генерируется диалоговое окно, показанное на рис. 1.25. Получив это сообщение, пользователь может проверить состояние принтера. Макрос приостанавливает работу до щелчка на кнопке OK.

```
MsgBox "Проверьте готовность принтера к печати"
```

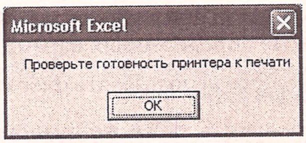


Рис. 1.25. Вывод предупреждения с помощью функции MsgBox

Если нужно поэкспериментировать, то для выполнения таких строк кода воспользуйтесь окном Immediate (Проверка). С другой стороны, код можно ввести в стандартный модуль в окне редактора VBE. В таком случае придется добавить ключевые слова Sub и End Sub, как показано ниже:

```
Sub Test1()
    MsgBox "Проверьте готовность принтера к печати"
End Sub
```

Для запуска подпрограммы щелкните на любой строке кода и нажмите клавишу <F5>.

Функция MsgBox принимает множество параметров, которые управляют типом кнопок и пиктограмм, доступных в диалоговом окне. Для получения справочной информации о функции MsgBox или о любом ключевом слове VBA поместите курсор на ключевое слово и нажмите клавишу <F1>. Сразу же после нажатия клавиши откроется окно справочного руководства, в котором будет показана информация о ключевом слове. Кроме всего прочего, в справочном руководстве указывается список доступных параметров функции:

```
MsgBox (prompt[, buttons] [, title] [, helpfile, context])
```

Параметры в квадратных скобках являются необязательными. В данном случае единственным обязательным параметром является сообщение. Если в заголовке диалогового окна должен выводиться текст, передайте его функции в качестве параметра. Значения параметров можно указывать как по позиции, так и по имени.

Передача параметров по позиции

Если указывать параметры по позиции, придется соблюдать правильный порядок параметров. Кроме этого, вместо отсутствующих параметров нужно указывать дополнительные запятые. Следующий код выводит диалоговое окно с заголовком, передавая параметр заголовка по позиции (рис. 1.26):

```
MsgBox "Принтер включен?", , "Внимание!"
```

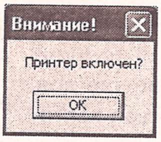


Рис. 1.26. Сообщение с заголовком

Передача параметров по имени

Передача параметров по имени обеспечивает несколько преимуществ:

- параметры могут вводиться в любом порядке и не нужно вводить дополнительные запятые для обозначения не определенных параметров;
- вместо оператора = необходимо использовать оператор :=, как было показано ранее.

Следующий код позволяет получить такое же диалоговое окно, как и в предыдущем примере:

```
MsgBox Title:= "Внимание!", Prompt:="Принтер включен?"
```

Еще одним преимуществом указания параметров по имени является большая документированность кода. Такая запись делает код намного понятнее.

Дополнительная информация о параметре `buttons` доступна в таблице параметров в справочном руководстве редактора VBE на странице функции `MsgBox`. Ниже показаны возможные значения этого параметра:

Константа	*Значение*	*Описание*
`VbOKOnly`	0	Отображает только кнопку OK
`VbOKCancel`	1	Отображает кнопки OK и Отмена (Cancel)
`VbAbortRetryIgnore`	2	Отображает кнопки Отмена (Abort), Повтор (Retry) и Игнорировать (Ignore)
`VbYesNoCancel`	3	Отображает кнопки Да (Yes), Нет (No) и Отмена (Cancel)
`VbYesNo`	4	Отображает кнопки Да (Yes) и Нет (No)
`VbRetryCancel`	5	Отображает кнопки Повтор (Retry) и Отмена (Cancel)
`VbCritical`	16	Отображает пиктограмму Критическая ошибка
`VbQuestion`	32	Отображает пиктограмму вопроса
`VbExclamation`	48	Отображает пиктограмму предупреждения

Константа	*Значение*	*Описание*
`VbInformation`	64	Отображает пиктограмму информационного сообщения
`VbDefaultButton1`	0	По умолчанию выделена первая кнопка
`VbDefaultButton2`	256	По умолчанию выделена вторая кнопка
`VbDefaultButton3`	512	По умолчанию выделена третья кнопка
`VbDefaultButton4`	768	По умолчанию выделена четвертая кнопка
`VbApplicatoinModal`	0	Модальное приложение. Пользователь должен ответить на сообщение перед тем, как продолжить работу с текущим приложением
`VbSystemModal`	4096	Модальная система. Все приложения приостанавливают работу, пока пользователь не ответит на сообщение
`VbMsgBoxHelpButton`	16384	Добавляет кнопку Справка (Help) в окно сообщения
`VbMsgBoxSetForeground`	65536	Выводит окно сообщения на передний план
`VbMsgBoxRight`	524288	Текст выравнивается по правому краю
`VbMsgBoxRtlReading`	1048576	Заставляет выводить текст справа налево, если используется еврейская или арабская система чтения

Значения от 0 до 5 управляют кнопками, которые выводятся в окне (рис. 1.27). Значение 4 заставляет окно отображать кнопки Да (Yes) и Нет (No):

```
MsgBox Prompt:="Удалить запись?", Buttons:=4
```

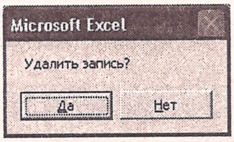


Рис. 1.27. Вывод сообщения с кнопками

Значения с 16 по 64 управляют пиктограммами, отображаемыми в окне сообщения (рис. 1.28). Значение 32 заставляет окно отображать пиктограмму со знаком вопроса. Если необходимо одновременно указать значения 4 и 32, их можно просто сложить:

```
MsgBox Prompt:="Удалить запись?", Buttons:=36
```

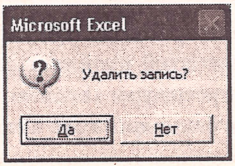


Рис. 1.28. Вывод сообщения с кнопками и пиктограммой

Константы

Если в качестве значения параметра Buttons указать 36, код будет никому не понятен, кроме самых закаленных программистов. Именно поэтому язык VBA предоставляет константы, показанные слева от значений на странице справочного руководства. Вместо указания числового значения параметра Buttons можно использовать символьные константы, которые более понятно описывают смысл использованного значения. Следующий код генерирует такое же диалоговое окно, как и в предыдущем примере:

```
MsgBox Prompt:="Удалить запись?", Buttons:=vbYesNo + vbQuestion
```

> Редактор VBE помогает набирать код, предоставляя всплывающий список соответствующих констант после ввода имени параметра (Buttons) и оператора :=. Выберите первую константу и нажмите клавишу <+>. Будет выдан список для выбора второй константы. Выберите вторую константу и нажмите пробел или <Tab> для завершения ввода строки. Если необходимо указать еще один параметр, введите "," и вводите имя следующего параметра.

Константы являются специальным типом переменных, который не поддерживает изменение значения. Константы используются для хранения ключевых данных и обеспечивают механизм создания более понятного кода. В языке VBA доступно множество встроенных констант. Программист может определить собственные константы, как показано далее в этой главе.

Возвращаемые значения

В примерах применения функции MsgBox кое-чего не хватает. Приложение задает вопрос, но не получает ответ пользователя. Это связано с тем, что MsgBox рассматривается как оператор, а не как функция. Такая интерпретация допустима, но для избежания синтаксических ошибок необходимо знать некоторые правила. Для получения возвращаемого значения функции MsgBox оно присваивается переменной.

Но если попытаться выполнить следующий код, будет выдано сообщение о синтаксической ошибке:

```
Answer = MsgBox Prompt:="Удалить запись?", Buttons:=vbYesNo + vbQuestion
```

Сообщение об ошибке Expected: End of Statement не проясняет ситуацию. Можно щелкнуть на кнопке Справка (Help) и получить более подробное описание, но даже этой информации может оказаться недостаточно для обнаружения источника ошибки.

Скобки

В приведенном выше коде проблема заключается в отсутствии скобок вокруг аргументов функции. Правильный код должен выглядеть следующим образом:

```
Answer = MsgBox(Prompt:="Удалить запись?", Buttons:=vbYesNo + vbQuestion)
```

Как правило, если необходимо получить возвращаемое значение функции, аргументы функции нужно заключить в скобки. Этой проблемы можно избежать, если всегда заключать в скобки аргументы вызова процедуры. Если возвращаемое значение не вызывает интерес, скобки можно не указывать или добавить в начале вызова ключевое слово Call, например:

```
Call MsgBox(Prompt:="Удалить запись?", Buttons:=vbYesNo + vbQuestion)
```

Правило скобок относится и к методам объектов. Многие методы возвращают значение, которое можно игнорировать или использовать. Соответствующий пример рассматривается в разделе, посвященном объектным переменным далее в этой главе.

Теперь, когда возвращаемое значение функции MsgBox сохраняется в переменной, его необходимо как-то использовать. И в этом случае дополнительная информация доступна в справочном руководстве в виде следующей таблицы:

Константа	*Значение*	*Описание*
VbOK	1	OK
VbCancel	2	Отмена (Cancel)
VbAbort	3	Прервать (Abort)
VbRetry	4	Повторить (Retry)
VbIgnore	5	Игнорировать (Ignore)
VbYes	6	Да (Yes)
VbNo	7	Нет (No)

Если щелкнуть на кнопке Да (Yes), функция вернет значение 6. В проверке If можно использовать как числовое значение, так и константу vbYes:

```
Answer = MsgBox(Prompt:="Удалить запись?", Buttons:=vbYesNo + vbQuestion)
If Answer = vbYes Than ActiveCell.EntireRow.Delete
...
```

Функция InputBox

Еще одна полезная функция VBA называется InputBox. Она позволяет получать от пользователя текстовые данные. Следующий код генерирует диалоговое окно, показанное на рис. 1.29:

```
UserName = InputBox(Prompt:="Введите имя пользователя")
```

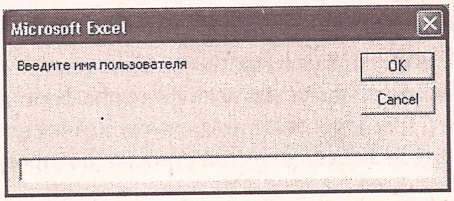


Рис. 1.29. Диалоговое окно с полем ввода

Результатом вызова функции InputBox является текстовая строка. Даже если ввести числовое значение, результат все равно останется текстовой строкой. Если щелкнуть на кнопке Отмена (Cancel) или OK, не вводя ничего в текстовое поле ввода, функция InputBox возвращает строку нулевой длины. Желательно проверять возвращаемое значение перед продолжением работы. В следующем примере подпрограмма не выполняет никаких действий при щелчке на кнопке Отмена (Cancel). Оператор Exit Sub завершает выполнение подпрограммы. В противном случае введенные данные копируются в ячейку B2:

```
Sub GetData()
    Sales = InputBox(Prompt:="Введите объем продаж")
    If Sales = "" Then Exit Sub
    Range("B2").Value = Sales
End Sub
```

В приведенном выше коде условие `If` сравнивает значение переменной `Sales` со строкой нулевой длины. Между двойными кавычками ничего нет. Не поддайтесь искушению ввести пробел между кавычками.

Существует более мощная версия функции `InputBox`*, которая является методом объекта* `Application`*. Эта функция позволяет ограничить тип вводимых данных. Она рассматривается в главе 3.*

Вызов функций и подпрограмм

При разработке приложения не пытайтесь разместить весь код в одной большой процедуре. Стоит писать небольшие процедуры, выполняющие конкретную задачу, что позволит проверять работоспособность каждой процедуры в отдельности. После этого можно создать главную процедуру, которая вызывает процедуры конкретных задач. Этот подход значительно упрощает тестирование и отладку приложения, а также его дальнейшую модификацию.

В следующем коде показан пример модульного подхода, хотя в реальном приложении процедуры будут содержать больший объем кода:

```
Sub Master()
    Dim SalesData As String
    SalesData = GetSalesData()
    If (SalesData <> "") Then
        Call PostInput(SalesData, "B3")
    End If
End Sub

Function GetSalesData()
    GetSalesData = InputBox("Введите объем продаж")
End Function

Sub PostInput(InputData, Target)
    Range(Target).Value = InputData
End Sub
```

Процедура `Master` использует функцию `GetSalesData` и подпрограмму `PostInput`. Функция `GetSalesData` выдает запрос, вызывая функцию `InputBox`, и возвращает введенные пользователем данные. Процедура `Master` проверяет ответ пользователя на неравенство пустой строке. Если значение не является пустой строкой, подпрограмма `PostInput` копирует данные на лист.

Обратите внимание, что параметры можно передавать как в подпрограммы, так и в функции. Но подпрограмму, принимающую параметры, нельзя запустить непосредственно с помощью нажатия клавиши <F5>.

Оператор Call

Ключевое слово `Call` является таким же старым, как и подпрограммы. На самом деле ключевое слово `Call` происходит из языка ассемблера, в котором применялась одноименная инструкция. Использование оператора `Call` и скобок позволяет явно указать, что код вызывает метод, и параметры являются аргументами вызова процедуры.

Объявление переменных

Ранее было показано множество примеров использования переменных для хранения информации. Пришло время рассмотреть различные типы переменных, наиболее подходящие способы их определения, а также правила создания имен переменных.

> Имена переменных могут состоять из букв и цифр, а также символа подчеркивания. Имя должно начинаться с буквы и может быть длиной до 255 символов. Желательно не использовать в именах переменных специальные символы, лучше только буквы алфавита (верхнего и нижнего регистров), а также цифры от 0 до 9 и символ подчеркивания (_). Кроме этого, в качестве имен переменных нельзя использовать ключевые слова VBA, например, `Sub` и `End`, а также названия функций VBA.

До этого момента переменные создавались через простое использование. Этот так называемое неявное объявление переменной. Большинство языков программирования требуют явного объявления переменных. Это значит, что перед применением в коде необходимо определить имена всех переменных. Язык VBA поддерживает оба типа объявлений. Для явного определения переменной используется оператор `Dim` или один из его вариантов, рассматриваемых далее. Следующий оператор `Dim` объявляет переменную, которая называется `SalesData`:

```
Sub GetData()
    Dim SalesData As String
    SalesData = InputBox(Prompt:="Введите объем продаж")
...
```

Неявное объявление переменных также поддерживается, но его стоит избегать. Код должен быть как можно более явным. Явный код и объявления переменных выглядят более осмысленно и делают код более точным. Кроме этого, неявные переменные получают инвариантный тип данных. Инвариантный тип данных требует дополнительной фоновой обработки для определения типа данных при каждом использовании переменной. Дополнительный код добавляется автоматически, и, кроме снижения ясности кода, инвариантные типы данных связаны с дополнительными накладными расходами во время выполнения.

Оператор Option Explicit

Существует метод принудительного включения явных объявлений в языке VBA. Для этого оператор `Option Explicit` вставляется в начало модуля, как показано на рис. 1.30.

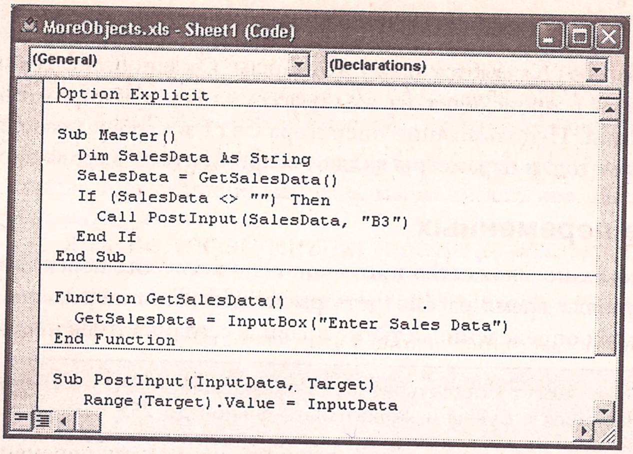


Рис. 1.30. Оператор Option Explicit указывается в начале модуля

> Оператор `Option Explicit` относится только к тому модулю, в котором он указан. Этот оператор должен быть добавлен в каждом модуле, требующем явного объявления переменных.

При попытке компиляции кода или запуска процедуры в модуле с включенным явным объявлением переменных интерпретатор VBA найдет все не объявленные переменные, выделит их другим цветом и выдаст сообщение об ошибке. Эта возможность VBA часто оказывается очень полезной. Рассмотрим следующую версию процедуры `GetSalesData`, в модуле которой не указывается оператор `Option Explicit`. В этом случае будет использоваться неявное объявление:

```
Sub GetData()
    SalesData = InputBox(Prompt:="Введите объем продаж")
    If SalesData = "" Then Exit Sub
    Range("B2").Value = SalesData
End Sub
```

Этот код никогда не введет данные в ячейку B2. Интерпретатор VBA примет неправильно введенное имя переменной `SaleData` в условном операторе `If` в качестве новой переменной, не имеющей значения. Для интерпретатора эта переменная будет эквивалентом пустой строки и всегда будет выполняться оператор `Exit Sub`, а последняя строка не будет выполняться никогда. Ошибки такого типа, особенно в коде большого объема, могут долгое время оставаться незамеченными.

Если указать оператор `Option Explicit` в начале раздела объявлений, а оператор `Dim SalesData` в начале подпрограммы `GetSalesData`, при попытке выполнения процедуры будет выдано сообщение об ошибке "Переменная не определена". Не определенная переменная будет выделена другим цветом, что позволит очень быстро обнаружить источник ошибки.

Оператор `Option Explicit` можно вставлять автоматически в каждый новый модуль. Для этого в редакторе VBE выберите команду Tools⇨Options (Сервис⇨Параметры) и активизируйте вкладку Editor (Редактор). Установите флажок Require Variable Declaration (Требовать объявления переменных), который всегда желательно устанавливать. Обратите внимание, что установка этого параметра не оказывает влияния на существующие модули, в которые оператор `Option Explicit` придется добавлять вручную.

Область видимости и время жизни переменных

С переменными связано две очень важных концепции:

- область видимости, определяющая возможность доступа к переменной;
- время жизни, определяющее, как долго эта переменная будет сохранять присвоенное значение.

В следующей процедуре приводится пример времени жизни переменной:

```
Sub LifeTime()
    Dim Sales As Integer
    Sales = Sales + 1
    Call MsgBox(Sales)
End Sub
```

При каждом запуске процедуры `LifeTime` выводимое значение переменной будет равно 1. Это связано с тем, что переменная `Sales` существует в памяти только до завершения работы процедуры. Память, выделенная для хранения переменной `Sales`, освобождается после достижения оператора `End Sub`. При следующем запуске процедуры `LifeTime` переменная `Sales` создается повторно и по умолчанию получает значение 0. Область видимости переменной `Sales` находится в пределах процедуры `LifeTime`, а время жизни переменной равно времени выполнения процедуры, в которой она определена. Для увеличения времени жизни переменной ее можно определить как статическую (ключевое слово `Static`):

```
Sub LifeTime()

    Static Sales As Integer

    Sales = Sales + 1
    MsgBox Sales
End Sub
```

Время жизни переменной `Sales` увеличивается до времени, в течение которого остается открытой книга. Значение переменной `Sales` увеличивается с каждым запуском процедуры `LifeTime`.

В следующих процедурах на примере показана область видимости переменной:

```
Sub Scope1()
    Static Sales As Integer
    Sales = Sales + 1
    MsgBox Sales
End Sub

Sub Scope2()
    Static Sales As Integer
    Sales = Sales + 10
    MsgBox Sales
End Sub
```

Переменная Sales в процедуре Scope1 не совпадает с переменной Sales в процедуре Scope2. При каждом запуске процедуры Scope1 значение переменной Sales увеличивается на единицу независимо от значения переменной Sales в подпрограмме Scope2. Точно так же переменная Sales в подпрограмме Scope2 будет увеличиваться на 10 при каждом вызове подпрограммы Scope2, независимо от переменной Sales в подпрограмме Scope1. Любая переменная, определенная внутри процедуры, имеет область видимости в пределах этой процедуры. Переменная, определенная в процедуре, называется переменной уровня процедуры.

Кроме этого, переменные могут определяться в разделе объявлений в начале модуля, как показано в следующем фрагменте кода:

```
Option Explicit
Dim Sales As Integer

Sub Scope1()
    Sales = Sales + 1
    MsgBox Sales
End Sub

Sub Scope2()
    Sales = Sales + 10
    MsgBox Sales
End Sub
```

В данном случае подпрограммы Scope1 и Scope2 работают с одной и той же переменной Sales. Переменная, которая определена в разделе объявлений модуля, называется переменной уровня модуля и область ее видимости совпадает с границами модуля. Таким образом, переменная остается видимой для всех процедур внутри модуля. Время жизни переменной соответствует времени, в течение которого остается открытой книга.

Если процедура внутри модуля объявляет переменную с тем же именем, что и у переменной уровня модуля, последняя станет невидимой для процедуры. В этом случае процедура будет работать с собственной переменной уровня процедуры.

Переменные уровня модуля, объявленные в разделе объявлений модуля с помощью оператора Dim, не видимы из других модулей. Если переменная должна использоваться одновременно в нескольких модулях, ее необходимо объявить с применением квалификатора Public:

```
Public Sales As Integer
```

Такие переменные можно сделать видимыми и в других книгах или проектах VBA. Для этого в другую книгу необходимо добавить ссылку на книгу, содержащую переменную Public. Для добавления ссылки можно воспользоваться командой меню Tools⇨References (Сервис⇨Ссылки) в редакторе VBE.

Тип переменной

Компьютеры используют различные методы хранения разных типов данных. Метод хранения чисел значительно отличается от метода хранения текстовых строк. Разные категории чисел также имеют разное представление в памяти. Целое число (без десятичной точки) хранится не так, как хранится число с десятичной точкой. Большинство языков требует объявления типа данных, для хранения которого будет использоваться переменная. В языке VBA такое требование не предъявляется, но код будет работать более эффективно, если типы переменных объявить заранее. Кроме этого, при раннем опре-

делении типа переменной упрощается обнаружение проблем, связанных с преобразованием данных из одного типа в другой.

Следующие таблицы были скопированы из справочного руководства по языку VBA. В таблицах определены различные типы данных, доступные в языке VBA, а также требования этих типов к объему памяти. Кроме этого, указывается диапазон значений, поддерживаемый каждым типом:

Тип данных	*Размер представления в памяти*	*Диапазон значений*
`Byte`	1 байт	от 0 до 255
`Boolean`	2 байта	True или False
`Integer`	2 байта	от –32768 до 32767
`Long` (длинное целое)	4 байта	от –2147483648 до 2147483647
`Single` (число с плавающей точкой одинарной точности)	4 байта	от –3.402823E38 до –1.401298E-45 для отрицательных значений и от 1.401298E-45 до 3.402823E38 для положительных значений
`Double` (число с плавающей точкой двойной точности)	8 байт	от –1.79769313486231E308 до –4.94065645841247E-324 для отрицательных значений и от 4.94065645841247E-324 до 1.79769313486231E308 для положительных значений
`Currency` (масштабированное целое)	8 байт	от -922337203685477.5808 до 922337203685477.5807
`Decimal`	14 байт	+/–79228162514264337593543950335 без десятичной точки, +/–7.9228162514264337593543950335 с 28 знаками после десятичной точки. Наименьшее ненулевое значение составляет +/–0.0000000000000000000000000001
`Date`	8 байт	от 1 января 100 года до 31 декабря 9999 года
`Object`	4 байта	ссылка на любой объект
`String` (переменная длина)	10 байт + длина строки	от 0 до примерно 2 миллиардов символов
`String` (фиксированная длина)	длина строки	от 1 до примерно 65400 символов
`Variant` (числовые данные)	16 байт	Любое числовое значение в пределах диапазона типа Double
`Variant` (символьные данные)	22 байт + длина строки	Тот же диапазон, что и тип String с переменной длиной строки
Определенный пользователем тип (определяется с помощью ключевого слова `Type`)	Объем, необходимый для хранения элементов	Диапазон каждого элемента совпадает с диапазоном типа данных

Если не объявить тип переменной, по умолчанию будет применен тип Variant. Этот тип данных использует больший объем памяти, так как вместе со значением переменной хранится информация о типе данных, которые в данный момент содержит переменная.

Использование типа Variant связано с дополнительными накладными расходами при обработке. Интерпретатор VBA должен определить, с каким типом данных приходится работать, а также необходимость его преобразования. Если от приложения требуется максимально возможная скорость работы, типы всех переменных необходимо определить явно. При этом применяйте типы, которые требуют минимального объема памяти для хранения значений. Например, если известно, что будут использоваться только целые числа в диапазоне от –32000 до 32000, стоит объявлять переменные типа Integer.

Определение типа переменной

Тип переменной можно определить с помощью оператора Dim с использованием дополнительных квалификаторов, например Public. В следующей строке переменная Sales объявляется как число с плавающей точкой и двойной точностью:

```
Dim Sales As Double
```

В одной строке можно объявить больше одной переменной:

```
Dim SalesData As Double, Index As Integer, StartDate As Date
```

Следующая строка может стать ловушкой для неопытного пользователя:

```
Dim Col, Row, Sheet As Integer
```

Многие пользователи предполагают, что эта строка объявляет все переменные, как имеющие тип Integer. Это не так. Переменные Col и Row получают тип Variant, так как явное объявление типа отсутствует. Для явного определения всех переменных необходимо воспользоваться следующим кодом:

```
Dim Col As Integer, Row As Integer, Sheet As Integer
```

Объявление функции и типы параметров

Если в подпрограмму или функцию передаются параметры, тип каждого из них можно определить в первой строке процедуры, например:

```
Function IsHoliday(WhichDay As Date)
```

```
Sub Marine(CrewSize As Integer, FuelCapacity As Double)
```

Кроме этого, можно определить тип возвращаемого значения функции. Функция из следующего примера возвращает значение True или False:

```
Function IsHoliday (WhichDay As Date) As Boolean
```

Константы

Ранее было показано, что в языке VBA предоставляется множество встроенных констант. Язык поддерживает определение собственных констант. Константы полезны для хранения чисел или текстовых строк, которые не меняются в процессе работы приложения, но часто используются в расчетах или сообщениях. Константы объявляются с помощью ключевого слова Const, как показано ниже:

```
Const Pi = 3.14159264358979
```

Но желательно указывать в объявлении и тип константы:

```
Const Version As String = "Release 3.9a"
```

Константы следуют тем же правилам определения области видимости, что и переменные. Если определить константу внутри процедуры, константа останется локальной для процедуры. Если определить константу в начале модуля, она будет доступна всем процедурам в пределах модуля. Если константа должна быть доступна всем модулям, ее можно объявить с квалификатором `Public`, например:

```
Public Const Error666 As String = "Мир больше не будет таким, как раньше"
```

Соглашения по именованию переменных

Самым важным правилом именования переменных в языке VBA (и в других языках программирования) является использование единого стиля именования. Многие программисты привыкли использовать префиксную запись, в которой несколько первых букв обозначают тип данных. Префиксная запись была разработана Чарльзом Симонием (Charles Symonyi) из компании Microsoft для слабо типизированного языка программирования С, и называлась Венгерской нотацией. (В сильно типизированных языках компилятор внимательно следит за соответствием объявленных типов аргументов и типов передаваемых в метод параметров.) Так как теперь большинство языков программирования обеспечивают более сильную типизацию, даже компания Microsoft рекомендует разработчикам применять явные объявления, полагаться на сильные типы и использовать контекст для описания назначения переменных, одновременно отказываясь от префиксной нотации.

Любой разработчик может применять собственное соглашение по именованию — компьютерам все равно, но людям плохо удается чтение и запоминание аббревиатур. Если в имя переменной необходимо включить тип, используйте полное имя класса, например `ButtonUpdateStatus` вместо `btnUpdateStatus`. С другой стороны, можно применять вариант `UpdateStatusButton`, который читается еще легче. В данном случае главное — сохранять единообразие при создании имен переменных.

Объектные переменные

Рассматриваемые до этого момента переменные хранили числа или текстовые строки. Кроме этого, существует возможность создания объектных переменных, которые ссылаются на объекты, например листы или диапазоны. Оператор `Set` используется для присвоения ссылки на объект объектной переменной. Как и обычные переменные, объектные переменные также требуют объявления и назначения типа. Если тип переменной не известен заранее, в объявлении можно указать универсальный тип `Object`:

```
Dim MyWorkbook As Object
Set MyWorkbook = ThisWorkbook
MsgBox MyWorkbook.Name
```

По возможности стоит использовать конкретный тип объекта. Следующий код создает объектную переменную `aRange`, ссылающуюся на ячейку B10 на листе `Sheet1` в той же книге, в которой хранится код. После этого объекту и ячейке над ним присваиваются новые значения:

```
Sub ObjectVariable()
    Dim aRange As Range
    Set aRange = ThisWorkbook.Worksheets("Sheet1").Range("B10")
    aRange.Value = InputBox("Введите объем продаж за январь")
    aRange.Offset(-1,0).Value = "January Sales"
End Sub
```

Если на один и тот же объект нужно ссылаться несколько раз, стоит объявить объектную переменную, а не несколько раз вводить длинную спецификацию объекта. Кроме этого, использование объектных переменных делает код проще для чтения и модификации.

Объектные переменные также полезны при сохранении возвращаемых значений некоторых методов. Особенно это касается методов создания новых экземпляров объектов. Например, метод `Add` объекта `Workbooks` или `Worksheets` возвращает ссылку на новый объект. Эта ссылка может быть присвоена объектной переменной, используемой для обращения к объекту в дальнейшем:

```
Sub NetWorkbook()
    Dim aWorkbook As Workbook, aWorksheet As Worksheet

    Set aWorkbook = Workbooks.Add
    Set aWorksheet = aWorkbook.Worksheets.Add ( _
    After:= aWorkbook.Sheets(aWorkbook.Sheets.Count))
    aWorksheet.Name = "Январь"
    aWorksheet.Range("A1").Value  = "Данные о продажах"
    aWorkbook.SaveAs Filename:="JanSales.xls"
End Sub
```

В этом примере создается новая пустая книга, ссылка на которую присваивается объектной переменной `aWorkbook`. В книгу после существующих листов добавляется новый лист, ссылка на который присваивается объектной переменной `aWorksheet`. Имя вкладки в нижней части листа меняется на "Январь", а в ячейку A1 вносится заголовок "Данные о продажах". Наконец, новая книга сохраняется в файле `JanSales.xls`.

Обратите внимание, что параметр вызова метода `Worksheets.Add` указан в скобках. Так как результат вызова метода `Add` присваивается объектной переменной, все параметры вызова должны указываться в скобках. Предпочтительней следовать этому соглашению. Если бы возвращаемое значение метода игнорировалось, скобки можно было бы не указывать:

```
Wkb.Worksheets.Add After:=Sheets(Wkb.Sheets.Count)
```

Но обычно стоит придерживаться единого стиля. Хорошей политикой является выбор удобного стиля кодирования с постоянным его использованием. (Помните, что очень немногие языки программирования поддерживают вызовы методов без скобок. Таким образом, начав программировать на другом языке, можно столкнуться с проблемами при попытке вызова метода без указания скобок. Так как придется программировать на нескольких языках, проще использовать скобки везде.)

Конструкция With... End With

Объектные переменные обеспечивают простой способ создания коротких ссылок на объекты. Кроме этого, объектные переменные обрабатываются в VBA более эффективно, чем полные ссылки на объекты. Еще одним способом сокращения объема кода и увеличения эффективности работы интерпретатора является использование структуры `With... End With`. Предыдущий пример можно переписать следующим образом:

```
With aWorkbook
     .Worksheets.Add (After:=.Sheets(.Sheets.Count))
End With
```

Интерпретатор VBA знает, что имя, начинающееся с точки, является именем свойства или метода объекта, указанного в операторе `With`. Можно переписать всю процедуру `NewWorkbook` и отказаться от использования объектной переменной `aWorkbook`:

```
Sub NewWorkbook()
    Dim aWorksheet As Worksheet
    With Workbook.Add
     Set aWorksheet = .Worksheets.Add(After:=.Sheets(.Sheets.Count))
     aWorksheet.Name = "Январь"
     aWorksheet.Range("A1").Value = "Данные о продажах"
     .SaveAs Filename:="JanSales.xls"
    End With
End Sub
```

Можно пойти еще дальше и полностью отказаться от использования объектной переменной Wks:

```
Sub NewWorkbook()
    With Workbooks.Add
     With .Worksheets.Add(After:=.Sheets(.Sheets.Count))
          .Name = "Январь"
          .Range("A1").Value = "Данные о продажах"
     End With
     .SaveAs Filename:="JanSales.xls"
    End With
End Sub
```

Если такая форма записи кажется запутанной, можно найти компромисс между использованием объектных переменных и конструкции With... End With:

```
Sub NewWorkbook()
Dim aWorkbook As Workbook, aWorksheet As Worksheet

Set aWorkbook = Workbooks.Add
    With aWorkbook
      Set aWorksheet = .Worksheets.Add(After:=.Sheets(.Sheets.Count))
      With aWorksheet
           .Name = "Январь"
           .Range("A1").Value = "Данные о продажах"
      End With
      .SaveAs FileName:="JanSales.xls"
     End With
End Sub
```

Конструкция With... End With особенно полезна, когда ссылки на объекты повторяются несколько раз в небольшом фрагменте кода.

Принятие решений

В языке VBA предоставляется две основных структуры для принятия решений и ветвления. Это операторы If и Select Case. Оператор If предоставляет большую гибкость, но Select Case удобнее при проверке значения единственной переменной.

Оператор If

Оператор If может принимать три формы: функция IIf, однострочный оператор If и блочный оператор If. В следующей функции Tax используется функция IIf (Immediate If):

```
Function Tas(ProfitBeforeTax As Double) As Double
    Tas = IIf(ProfitBeforeTax > 0, 0.3 * ProfitBeforeTax, 0)
End Function
```

Функция IIf похожа на тернарный оператор ?: из языков C, C++ и C#. Этот оператор имеет форму проверка?истина:ложь и похож на функцию IF для листов Excel.

Функция IIf принимает три аргумента: первый является логическим условием, второй — выражением, которое вычисляется при истинности условия, а третий — выражением, вычисляемым при ложности условия. В этом примере функция IIf сравнивает значение переменной ProfitBeforeTax с 0. Если условие истинно, функция IIf вычисляет 30% от значения переменно ProfitBeforeTax. Если условие ложно, функция IIf возвращает 0. Вычисленное значение функции IIf передается как возвращаемое значение функции Tax. Функцию Tax можно переписать для использования однострочного оператора If:

```
Function Tax(ProfitBeforeTax As Double) As Double
    If ProfitBeforeTax > 0 Then Tax = 0.3 *ProfitBeforeTax
↳ Else Tax = 0
End Function
```

Единственной разницей между функцией IIf и оператором If является необязательность раздела Else в однострочном варианте оператора If. Функция IIf требует определения всех трех параметров, в то время как в VBA часто бывает полезно опустить раздел Else:

```
If ProfitBeforeTax < 0 Then MsgBox "Возник убыток", ,"Предупреждение"
```

Еще одно отличие — это способность функции IIf присваивать значение только одной переменной, а однострочный оператор If может присваивать значения разным переменным:

```
If JohnsScore > MarysScore Then John = John + 1 Else Mary = Mary + 1
```

Блочный оператор If

Если при истинности условия необходимо выполнить больше одного действия, можно воспользоваться блочным оператором If, который показан ниже:

```
If JohnsScore > MarysScore Then
   John = John + 1
   Mary = Mary - 1
End If
```

При применении блочного оператора If код должен вводиться в следующей строке, а не после ключевого слова Then. После строки с условием можно указывать любое количество строк кода. Область видимости оператора If необходимо завершить ключевым словом End If. В блочном операторе If можно использовать раздел Else, например:

```
If JohnsScore > MarysScore Then
   John = John + 1
   Mary = Mary - 1
Else
   John = John - 1
   Mary = Mary + 1
End If
```

Кроме этого, в блочном операторе If можно использовать любое количество разделов ElseIf:

```
If JohnsScore > MarysScore Then
   John = John + 1
   Mary = Mary - 1
ElseIf JohnsScore < MarysScore Then
   John = John - 1
   Mary = Mary + 1
```

```
Else
   John = John + 1
   Mary = Mary + 1
End If
```

При применении блочного оператора If с одним или несколькими разделами ElseIf интерпретатор VBA продолжает проверку условий, пока не будет найден раздел с истинным условием. Сначала выполняется код из этого раздела, после чего выполнение продолжается с оператора, который следует за ключевым словом End If. Если ни одно из условий не является истинным, выполняется код из раздела Else.

Блочный оператор If не выполняет никаких действий, если ни одно из условий не является истинным и раздел Else отсутствует. Блочные операторы If можно вкладывать друг в друга. Стоит использовать выравнивание, чтобы обозначить область видимости каждого оператора. Это очень важно, так как очень легко запутаться во вложенных блоках If внутри других блоков If и блоках If внутри блоков Else. Если при написании кода выравнивание не используется, сопоставление каждого If с соответствующим End If превращается в сложную задачу:

```
If Not ThisWorkbook.Saved Then
   Answer = MsgBox("Сохранить изменения", vbQuestion + vbYesNo)
   If Answer = vbYes Then
      ThisWorkbook.Save
      MsgBox ThisWorkbook.Name & " сохранена"
   End If
End If
```

В этом коде применяется свойство Saved объекта Workbook, в котором хранится код. Свойство показывает, сохранялась ли книга после внесения изменений. Если изменения не сохранены, у пользователя запрашивается разрешение на их сохранение. Если пользователь утвердительно отвечает на запрос, внутренний блок If сохраняет книгу и сообщает об этом пользователю.

Оператор Select Case

Следующий блочный оператор If проверяет значение одной и той же переменной в каждом разделе:

```
Function Price(Product As String) As Variant
  If Product = "Apples" Then
    Price = 12.5
  ElseIf Product = "Апельсины" Then
    Price = 15
  ElseIf Product = "Груши" Then
    Price = 18
  ElseIf Product = "Манго" Then
    Price = 25
  Else
    Price = CVErr(xlErrNA)
  End If
End Function
```

Если соответствующее значение переменной Product не найдено, функция Price возвращает сообщение об ошибке Excel (#NA). Обратите внимание, что возвращаемое значение функции Price по определению имеет тип Variant. Это значит, что функция может возвращать как числовое значение, так и значение сообщения об ошибке. В такой ситуации более удобным решением было бы использование конструкции Select Case. В данном случае эта конструкция выглядела бы следующим образом:

```
Function Price(Product As String) As Variant
  Select Case Product
    Case "Яблоки"
      Price = 12.5
    Case "Апельсины"
      Price = 15
    Case "Груши"
      Price = 18
    Case "Манго"
      Price = 25
    Case Else
      Price = CVErr(xlErrNA)
  End Select
End Function
```

Если для каждого случая выполняется только один оператор, можно воспользоваться следующим форматом конструкции. Для указания нескольких операторов в одной строке между ними необходимо вставлять точку с запятой (;).

```
Function Price(Product As String) As Variant
  Select Case Product
    Case "Яблоки":  Price = 12.5
    Case "Апельсины": Price = 15
    Case "Груши":   Price = 18
    Case "Манго": Price = 25
    Case Else:      Price = CVErr(xlErrNA)
  End Select
End Function
```

Конструкция `Select Case` также позволяет работать с диапазонами чисел или с текстом. Кроме этого, в условии сравнения можно использовать ключевое слово `Is`. В следующем примере рассчитывается стоимость проезда для детей до 3 лет, для лиц старше 65 лет, а также для лиц, попадающих в промежуточную возрастную категорию. Если в качестве параметра функции указать отрицательный возраст, будет выдано сообщение об ошибке.

```
Function Fare(Age As Integer) As Variant
  Select Case Age
    Case 0 To 3, Is > 65
      Fare = 0
    Case 4 To 15
      Fare = 10
    Case 16 To 65
      Fare = 20
    Case Else
      Fare = CVErr(xlErrNA)
  End Select
End Function
```

Циклы

Во всех языках программирования предоставляются эффективные механизмы для многократного повторения одних и тех же или похожих последовательностей операций. В языке VBA даются четыре конструкции, позволяющие циклически выполнять один и тот же код. Это конструкции `While... Wend`, `Do... Loop`, `For... Next` и `For Each`.

В конструкциях `While...Wend` и `Do... Loop` условие указывается в начале цикла. В конструкции `While...Wend` условие указывается в начале цикла и тело цикла может выполняться 0 или больше раз. В конструкции `Do... Loop` условие можно указывать как в начале, так и в конце цикла. Если условие указывается в конце цикла, тело цикла будет

выполняться как минимум один раз. Цикл `For... Next` обычно используется, когда количество итераций цикла известно заранее, например, при переборе массива из 50 элементов. Цикл `For Each` используется для обработки объектов, входящих в коллекцию. Пример каждого цикла приводится в следующих разделах.

Цикл While...Wend

Цикл `While...Wend` имеет следующий синтаксис: `While(условие)...Wend`. Тело цикла указывается между операторами `While` и `Wend`. Пример цикла `While...Wend` показан ниже. Этот код меняет цвет фона ячеек в каждой строке.

```
Public Sub ShadeEverySecondRowWhileWend()
  Range("A2").EntireRow.Select
  While ActiveCell.Value <> ""
    Selection.Interior.ColorIndex = 10
    ActiveCell.Offset(2, 0).EntireRow.Select
  Wend
End Sub
```

Цикл Do...Loop

Для демонстрации работы цикла `Do...Loop` создадим процедуру, которая закрашивает каждую вторую строку листа, как показано на рис. 1.31. Макрос должен работать на разных листах отчетов с разным количеством продуктов, поэтому макрос проверяет столбец A в каждой строке, пока не встретит строку с пустой ячейкой в столбце A. Это и будет условием завершения цикла.

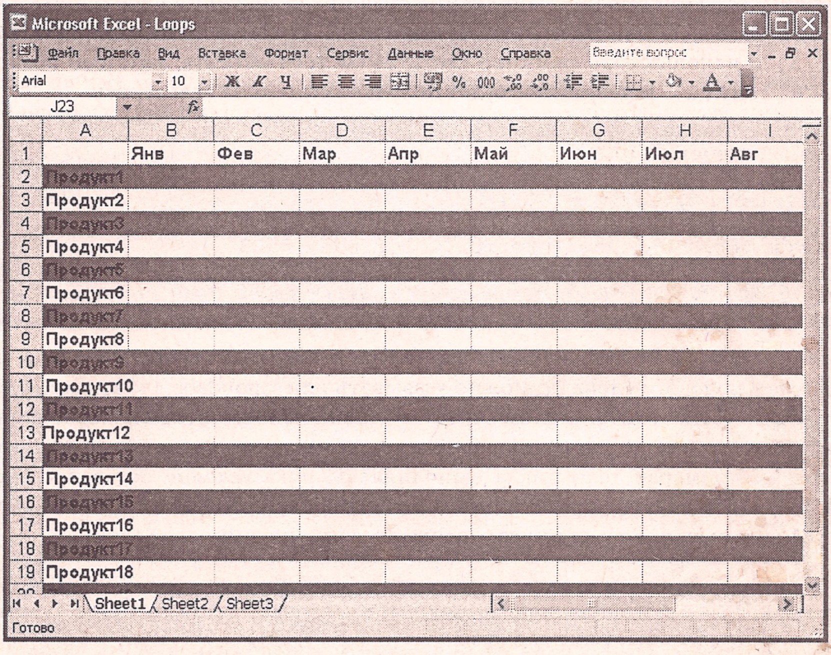


Рис. 1.31. Форматирование ячеек с помощью цикла

Первый макрос будет выбирать каждую вторую строку и применять соответствующее форматирование:

```
Public Sub ShadeEverySecondRow()
  Range("A2").EntireRow.Select
  Do While ActiveCell.Value <> ""
    Selection.Interior.ColorIndex = 15
    ActiveCell.Offset(2, 0).EntireRow.Select
  Loop
End Sub
```

Подпрограмма ShadeEverySecondRow начинает работу со второй строки. После этого она выделяет всю строку, и самая левая ячейка (А) становится активной ячейкой. Код между операторами Do и Loop выполняется до тех пор, пока свойство Value активной ячейки не станет равно пустой строке, то есть, пока активная ячейка содержит данные. В цикле макрос устанавливает индекс цвета фона выделенной ячейки равным 15 (серый цвет). После этого макрос выделяет всю строку на две строки ниже активной ячейки. После выделения строки, в которой ячейка А не содержит данных, условие While становится ложным и выполнение цикла завершается.

Процедуру ShadeEverySecondRow можно ускорить, если отказаться от выделения. В VBA очень редко требуется выделение ячеек, но здесь этот подход использовался из-за того, что именно так эти операции выполняются вручную и такой результат получается в процессе работы механизма записи макросов.

В следующей версии процедуры ShadeEverySecondRow ячейки не выделяются и процедура работает примерно в шесть раз быстрее. В процедуре определяется переменная индекса i, указывающая на строку листа. Изначально переменной присваивается значение 2. Свойство Cells объекта листа позволяет ссылаться на ячейки по номеру строки и номеру столбца, поэтому в начале работы цикла оператор Cells(i,1) указывает на ячейку A2. При каждом выполнении тела цикла значение переменной i увеличивается на 2. Таким образом, ссылку на активную ячейку можно заменить на ссылку Cells(i,1) и использовать свойство EntireRow того объекта, который возвращается вызовом Cells(i,1). В результате будет получена ссылка на всю строку:

```
Public Sub ShadeEverySecondRow()
  Dim i As Integer
  i = 2
  Do Until IsEmpty(Cells(i, 1))
    ' контрастный цвет
    Cells(i, 1).EntireRow.Interior.ColorIndex = 23
    i = i + 2
  Loop
End Sub
```

Для демонстрации альтернативных вариантов в оператор Do были внесены два изменения. После ключевого слова Do можно указывать или ключевое слово While, или ключевое слово Until, поэтому в данном случае в условии используется ключевое слово Until и функция VBA IsEmpty, которая позволяет проверить, не является ли ячейка пустой. В предыдущих примерах тело цикла выполнялось, пока условие оставалось истинным (равно True). В этом случае при использовании ключевого слова Until цикл выполняется, пока условие ложно. Работа цикла завершается при истинности условия.

> Функция IsEmpty является самым лучшим способом проверки пустоты ячейки. Условие If Cells(i,1) = "" будет выполняться для всех ячеек, в которых формулы возвращают пустую строку в качестве значения.

Кроме этого, существует возможность завершения цикла с помощью условия в теле цикла и оператора Exit Do. Такая конструкция показана в следующем примере. Также в примере показан еще один способ получения ссылки на всю строку.

```
Public Sub ShadeEverySecondRow()
  i = 0
  Do
    i = i + 2
    If IsEmpty(Cells(i, 1)) Then Exit Do
    Rows(i).Interior.ColorIndex = 15
  Loop
End Sub
```

Еще один вариант подразумевает добавление ключевого слова `While` или `Until` в строке с ключевым словом `Loop`. Это позволяет обеспечить как минимум одно выполнение тела цикла. Если условие указывается в строке с оператором `Do`, оно может быть ложным с самого начала и тело цикла не будет выполнено ни разу.

Иногда условие стоит проверять в последней строке цикла. В следующем примере видно, что проверка правильности ввода пароля (`PassWord`) имеет смысл только после получения ввода от пользователя, хотя код будет работать даже в том случае, если условие `Until` указать в строке `Do`:

```
Public Sub GetPassword()
  Dim PassWord As String, i As Integer
  i = 0
  Do     i = i + 1
    i = i + 1
    If i > 3 Then
      MsgBox "Извините, только три попытки"
      Exit Sub
    End If
    PassWord = InputBox("Введите пароль")
    Loop Until PassWord = "XXX"
  MsgBox "Добро пожаловать"
End Sub
```

Подпрограмма `GetPassword` выполняет тело цикла, пока не будет введен пароль XXX, но не более трех раз.

Цикл For... Next

Цикл `For... Next` отличается от цикла `Do... Loop` в двух аспектах. В цикле `For... Next` используется встроенный счетчик, который автоматически увеличивается на каждой итерации цикла, и цикл завершает работу, когда значение счетчика превысит указанное пороговое значение. То есть, количество итераций не зависит от указанного пользователем логического условия. В следующем примере полный путь и имя книги указываются в центре нижнего колонтитула каждого листа активной книги:

```
Public Sub FilePathInFooter()
  Dim i As Integer, FilePath As String

  FilePath = ActiveWorkbook.FullName
  For i = 1 To Worksheets.Count Step 1
    Worksheets(i).PageSetup.CenterFooter = FilePath
  Next i
End Sub
```

В версиях Excel до Excel 2003 отсутствовала возможность автоматического включения полного пути файла в верхний или нижний колонтитулы, поэтому такой макрос можно использовать для реализации недостающей функциональности. Вначале макрос присваивает значение свойства `FullName` объекта активной книги переменной `FilePath`. Цикл начинается с оператора `For` и выполняется до оператора `Next`. Пере-

менная i используется в качестве счетчика начиная с 1 и заканчивая значением свойства Worksheets.Count. Свойство Count коллекции Worksheets применяется для определения количества листов в активной книге.

Параметр Step определяет величину прироста счетчика на каждой итерации цикла. По умолчанию счетчик цикла For... Next увеличивается на 1. Можно использовать любое положительное или отрицательное значение прироста, например 2, 3 или -1.5. В этом примере переменная счетчика i используется в качестве индекса коллекции Worksheets для перебора отдельных объектов Worksheet. Свойство PageSetup объекта Worksheet ссылается на объект PageSetup, содержащий параметры печати данного листа. Свойству CenterFooter объекта PageSetup присваивается значение переменной FilePath.

В следующем примере показано, как реализовать обратный перебор. В данном случае из полного пути извлекается имя файла без расширения. В этом примере в качестве исходных данных используется значение свойства FullName активной книги, но этот же код можно применять для файлов любого типа. Код начинает работу с последнего символа имени файла и перемещается к началу, пока не найдет точку между именем файла и расширением. После этого код находит обратную косую черту между именем каталога и именем файла и извлекает имя файла между косой чертой и точкой:

```
Public Sub GetFileName()
  Dim BackSlash As Integer, Point As Integer
  Dim FilePath As String, FileName As String
  Dim i As Integer
  FilePath = ActiveWorkbook.FullName
  For i = Len(FilePath) To 1 Step -1
    If Mid$(FilePath, i, 1) = "." Then
      Point = i
      Exit For
    End If
  Next i
  If Point = 0 Then Point = Len(FilePath) + 1
  For i = Point - 1 To 1 Step -1
    If Mid$(FilePath, i, 1) = "\" Then
      BackSlash = i
      Exit For
    End If
  Next i
  FileName = Mid$(FilePath, BackSlash + 1, Point - BackSlash - 1)
  MsgBox FileName
End Sub
```

В первом цикле For... Next для определения количества символов в значении переменной FilePath используется функция Len. После этого счетчик i настраивается на обратный перебор начиная с позиции последнего символа в сторону начала строки. Функция Mid$ извлекает символ строки FilePath в позиции i, после чего символ сравнивается с точкой.

После обнаружения точки в имени файла ее позиция заносится в переменную Point, и первый цикл For... Loop завершает свою работу. Если в имени файла отсутствует расширение, точка не будет найдена и переменная Point будет иметь принятое по умолчанию значение 0. В таком случае условие If присвоит переменной Point мнимую позицию точки, которая находится на один символ правее последнего символа имени файла.

Этот же прием используется во втором цикле For... Next. Начиная со следующего слева символа после точки, цикл перебирает имя файла, пока не обнаружит обратную косую черту. Позиция обратной косой черты заносится в переменную BackSlash. После этого символы между точкой и обратной косой чертой извлекаются из имени файла с помощью функции Mid$.

Цикл For Each... Next

Если приходится обрабатывать каждый член коллекции, можно воспользоваться циклом `For Each... Next`. Следующий пример является переработкой процедуры `FilePathInFooter`:

```
Public Sub FilePathInFooter()
  Dim FilePath As String, aWorksheet As Worksheet

  FilePath = ActiveWorkbook.FullName
  For Each aWorksheet In Worksheets
    aWorksheet.PageSetup.CenterFooter = FilePath
  Next aWorksheet
End Sub
```

Тело цикла перебирает всех членов коллекции. На каждой итерации ссылка на следующего члена присваивается объектной переменной `aWorksheet`.

В следующем примере выводится список файлов корневого каталога на диске С. В этом случае используется объект Microsoft Office `FileSearch`, позволяющий сгенерировать объект `FoundFiles`, в котором содержатся имена найденных файлов. В данном случае для вывода имен файлов используется цикл `For Each... Next`.

```
Public Sub FileList()
  Dim File As Variant
  With Application.FileSearch
    .LookIn = "C:\"
    .FileType = msoFileTypeAllFiles
    .Execute
    For Each File In .FoundFiles
      MsgBox File
    Next File
  End With
End Sub
```

Если работа процедуры проверяется на каталоге с большим количеством файлов, необходимость постоянно щелкать на кнопке OK может утомить. В таком случае можно воспользоваться механизмом прерывания работы подпрограммы, который вызывается комбинацией клавиш <Ctrl+Break>.

Массивы

Массивы представляют собой переменные VBA, в которых хранится больше одного элемента данных. Массив объявляется при помощи скобок, указываемых после его имени. В скобках указывается целое число, которое определяет количество элементов массива:

```
Dim Data(2) As Integer
```

Для присвоения значений элементам массива необходимо указать номер элемента, как показано ниже:

```
Data(0) = 1
Data(1) = 10
Data(2) = 100
```

Количество элементов в массиве зависит от базового индекса массива. Принятый по умолчанию базовый индекс равен 0. Это значит, что первый элемент массива имеет номер 0. Если базовый индекс равен 0, оператор `Dim Data(2) As Integer` объявляет массив из трех целых элементов. В то же время в разделе объявлений модуля можно добавить следующий оператор, который сделает базовый индекс массива равным 1:

```
Option Base 1
```

При использовании единичного базового индекса оператор Dim Data(2) As Integer будет объявлять массив из двух целых элементов. Элемент с индексом 0 не будет существовать.

Для проверки эффекта оператора Option Base можно воспользоваться следующей процедурой:

```
Public Sub Array1()
  Dim Data(10) As Integer
  Dim Message As String, i As Integer
  For i = LBound(Data) To UBound(Data)
    Data(i) = i
  Next i
  Message = "Нижняя граница = " & LBound(Data) & vbCr
  Message = Message & "Верхняя граница = " & UBound(Data) & vbCr
  Message = Message & "Количество элементов = " &
↳ WorksheetFunction.Count(Data) & vbCr
  Message = Message & "Сумма элементов = " &
↳ WorksheetFunction.Sum(Data)
  MsgBox Message
End Sub
```

В массиве Array1 используются функции LBound (нижняя граница) и UBound (верхняя граница) для определения наименьшего и наибольшего значения индекса массива. Функция листа Count применяется для определения количества элементов массива. Если запустить этот код после выполнения оператора Option Base 0 или без оператора Option Base, наименьшее значение индекса массива будет равно 0, а в массиве будет 11 элементов. При использовании оператора Option Base 1 наименьший индекс составит 1 и в массиве будет храниться 10 элементов.

Обратите внимание на использование встроенной константы vbCr, в которой хранится символ возврата каретки. Константа vbCr применяется для разбивки текста сообщения на несколько строк.

Если размер массива не должен зависеть от оператора Option Base, нижнее и верхнее значения индекса можно определить явным образом, например:

```
Dim Data(1 to 2) As Integer
```

Массивы исключительно полезны для обработки групп элементов. Если необходимо создать короткий список, воспользуйтесь функцией Array, как показано ниже:

```
Dim Data As Variant
Data = Array ("Север", "Юг", "Восток", "Запад")
```

После этого список можно использовать в цикле For... Next. Например, можно открыть и обработать последовательность книг, которые называются Север.xls, Юг.xls, Восток.xls и Запад.xls:

```
Sub Array2()
  Dim Data As Variant, aWorkbook As Workbook
  Dim i As Integer

  Data = Array("Север", "Юг", "Восток", "Запад")
  For i = LBound(Data) To UBound(Data)
    Set aWorkbook = Workbooks.Open(FileName:=Data(i) & ".xls")
    ' Здесь обрабатываются данные
    aWorkbook.Close SaveChanges:=True
  Next i
End Sub
```

Многомерные массивы

До этого момента рассматривались массивы только с одним измерением. На самом деле можно определять массивы с несколькими измерениями (до 60). Хотя компьютеры с легкостью справляются со сложностью n-мерных массивов, где n больше 4 или 5, люди практически не в состоянии воспринимать такие конструкции. Следующие операторы определяют двумерные массивы:

```
Dim Data(10,20) As Integer
Dim Data(1 To 10, 1 To 20) As Integer
```

Двумерный массив можно представить в виде таблицы с данными. В предыдущем примере определяется таблица с 10 строками и 20 столбцами.

Массивы в Excel очень удобны для обработки данных из диапазонов на листах. Загрузка данных из листа в массив, обработка и запись результата обратно на лист оказывается намного эффективнее, чем обработка каждой ячейки в отдельности.

В следующей процедуре показано, как значения из диапазона присвоить переменной типа `Variant`. В этом коде функции `LBound` и `UBound` используются для определения размерности массива `Data`. Обратите внимание, что функции `LBound` и `UBound` принимают второй параметр, который указывает интересующий индекс. Если этот параметр не указывать, функция будет рассматривать первый индекс:

```
Public Sub Array3()
  Dim Data As Variant, X As Variant
  Dim Message As String, i As Integer

  Data = Range("A1:A20").Value
  i = 1
  Do
    Message = "Нижняя граница = " & LBound(Data, i) & vbCr
    Message = Message & "Верхняя граница = " & UBound(Data, i) & vbCr
    MsgBox Message, , "Индекс = " & i
    i = i + 1
    On Error Resume Next
    X = UBound(Data, i)
    If Err.Number <> 0 Then Exit Do
    On Error GoTo 0
  Loop
  Message = "Количество непустых элементов = " _
    & WorksheetFunction.CountA(Data) & vbCr
  MsgBox Message
End Sub
```

При первом выполнении цикла `Do... Loop` подпрограмма `Array3` определяет верхнюю и нижнюю границы первого индекса массива `Data`, так как переменная `i` имеет значение 1. После этого значение переменной `i` увеличивается на единицу для просмотра второго индекса. Работа цикла завершается в случае сообщения об отсутствии дополнительных измерений.

Передавая в подпрограмму `Array3` различные диапазоны, можно увидеть, что при назначении диапазона значений переменной типа `Variant` создается двумерный массив. Такой массив создается даже в том случае, если в диапазоне присутствует только одна строка или один столбец. Кроме этого, оказывается, что нижняя граница индекса всегда равна 1, вне зависимости от оператора `Option Base` в начале модуля.

Динамические массивы

При создании кода не всегда можно заранее определить размер массива, который потребуется в процессе работы. Например, может понадобиться загрузить в массив имена всех файлов с расширением `.xls` из текущего каталога. Одним из вариантов является объявление достаточно большого массива, который вместит максимально возможное количество элементов. Но это решение отличается неэффективностью. Вместо этого можно определить динамический массив, размер которого устанавливается в процессе работы процедуры.

Для определения динамического массива достаточно не указывать размерность в объявлении:

```
Dim Data() As String
```

Для указания необходимого размера в процессе работы воспользуйтесь оператором `ReDim`. При этом размер массива можно передавать в виде значений переменных:

```
ReDim Data(iRows, iColumns) As String
ReDim Data(minRow to maxRow, minCol to maxCol) As String
```

Использование оператора `ReDim` приводит к переинициализации массива и уничтожению всех хранящихся в массиве данных. Для сохранения данных необходимо применять ключевое слово `Preserve`. Это ключевое слово используется в следующей процедуре. В этом примере имена файлов загружаются в динамический массив `FNames` в теле цикла `Do... Loop`. При этом на каждой итерации верхняя граница индекса массива увеличивается на единицу, чтобы обеспечить место для хранения нового имени. Функция `Dir` возвращает первое имя, соответствующее шаблону из переменной `FType`. Последующие вызовы функции `Dir` без параметра приводят к использованию того же шаблона, но функция возвращает следующий файл, соответствующий спецификации. После возврата всех имен файлов функция возвращает пустую строку.

```
Public Sub FileNames()
  Dim FName As String
  Dim FNames() As String
  Dim FType As String
  Dim i As Integer

  FType = "*.xls"
  FName = Dir(FType)
  Do Until FName = ""
    i = i + 1
    ReDim Preserve FNames(1 To i)
    FNames(i) = FName
    FName = Dir
  Loop
  If i = 0 Then
    MsgBox "Файлы не найдены"
  Else
    For i = 1 To UBound(FNames)
      MsgBox FNames(i)
    Next i
  End If
End Sub
```

Если планируется обработка файлов из каталога с последующим сохранением результатов, желательно сначала получить весь список файлов (как в процедуре `FileNames`), после чего использовать этот список для обработки. Если приложение постоянно читает и перезаписывает файлы в каталоге, на функцию `Dir` полагаться не стоит.

Обработка ошибок на этапе выполнения

При проектировании приложения необходимо предусмотреть возникновение всех проблем, возможных при использовании приложения в реальных условиях. Можно исправить все ошибки и спроектировать безупречную логику, которая срабатывает при всех наборах условий, но простая проблема в работе аппаратных средств может привести к аварийному завершению работы кода с невразумительным сообщением об ошибке.

Например, если попытаться сохранить файл книги на гибком диске в приводе A, когда гибкий диск в приводе отсутствует, код остановит свою работу и выдаст сообщение, которое не каждый пользователь сможет расшифровать.

Если есть вероятность появления такой ошибки, код может незаметно обработать такую ситуацию. Для этого язык VBA предоставляет механизм перехвата ошибочных состояний с помощью следующего оператора:

```
On Error GoTo LineLabel
```

`LineLabel` является меткой, добавляемой в конец нормального кода, как показано ниже (метка `errorTrap`). Обратите внимание, что после метки указывается символ двоеточия. Метка строки обозначает начало кода обработки ошибки. Перед меткой необходимо добавить оператор `Exit`, который предотвратит выполнение кода обработки ошибки после нормального завершения работы кода:

```
Public Sub ErrorTrap1()
  Dim Answer As Long, MyFile As String
  Dim Message As String, CurrentPath As String

  On Error GoTo errorTrap
  CurrentPath = CurDir$

  ' Попытаться сменить диск на диск A:

  ChDrive "A"
  ChDrive CurrentPath
  ChDir CurrentPath
  MyFile = "A:\Data.xls"
  Application.DisplayAlerts = False
  ` Попытаться сохранить лист на диск A:
  ActiveWorkbook.SaveAs FileName:=MyFile

TidyUp:
  ChDrive CurrentPath
  ChDir CurrentPath
Exit Sub

errorTrap:
  Message = "Номер ошибки: = " & Err.Number & vbCr
  Message = Message & Err.Description & vbCr & vbCr
  Message = Message & "Вставьте диск в привод A:" & vbCr
  Message = Message & "и щелкните на кнопке OK" & vbCr & vbCr
  Message = Message & "Щелкните на кнопке Отмена для отмены
↳ сохранения"
  Answer = MsgBox(Message, vbQuestion + vbOKCancel, "Error")
  If Answer = vbCancel Then Resume TidyUp
  Resume
End Sub
```

После выполнения оператора `On Error` механизм перехвата ошибок оказывается включен. При возникновении ошибки не выдаются никакие сообщения и выполняется код, который находится после соответствующей метки. С помощью объекта `Err` можно полу-

чить информацию о возникшей ошибке. Свойство `Number` объекта `Err` возвращает номер ошибки, а в свойстве `Description` предоставляется сообщение, связанное с этой ошибкой. Свойство `Err.Number` можно использовать для определения возникшей ошибки при наличии нескольких различных ошибок. Содержимое свойства `Err.Description` можно использовать в собственном сообщении об ошибке.

> *В Excel 5 и Excel 95 конструкция `Err` представляла собой не объект, а функцию, которая возвращала номер ошибки. Так как свойство `Number` является принятым по умолчанию свойством объекта `Err`, использование `Err` эквивалентно использованию `Err.Number`. Это позволяет коду из более старых версий Excel без изменений работать в Excel 97 и более новых версиях.*

После оператора `On Error` код процедуры `ErrorTrap1` сохраняет текущий путь и имя диска в переменной `CurrentPath`. Потом выполняется оператор `ChDrive`, который пытается активизировать привод А. Если в приводе отсутствует гибкий диск, возникает ошибка `68 (Device unavailable)` и выполняется код обработки ошибки. В демонстрационных целях код ошибки и описание выводятся на экран, и пользователю предоставляется возможность вставить диск в привод и продолжить работу или отказаться от сохранения.

Если пользователь выбирает завершение работы, управление передается обратно на метку `TidyUp` и процедура восстанавливает начальные параметры диска и каталога. В противном случае выполняется оператор `Resume`, который передает управление оператору, выполнение которого привело к появлению ошибки. Если в приводе А до сих пор нет диска, код обработки ошибки будет выполнен еще раз. В противном случае код продолжит нормальную работу.

Единственным назначением оператора `ChDrive "A"` является проверка готовности привода А, поэтому код восстанавливает параметры диска и каталога. Перед сохранением активной книги код устанавливает свойство `DisplayAlerts` объекта `Application` в значение `False`. Это позволяет скрыть предупреждение о перезаписи старого файла `Data.xls` новым. (Дополнительная информация о свойстве `DisplayAlerts` приводится в главе 18.)

Оператор `Resume` доступен в трех вариантах:

- `Resume` — приводит к выполнению оператора, который привел к появлению ошибки;
- `Resume Next` — передает управление оператору, следующему после оператора, который привел к появлению ошибки;
- `Resume LineLabel` — передает управление на произвольную метку строки. Это позволяет продолжить выполнение кода в любом месте.

В следующем коде оператор `Resume Next` используется для обхода оператора `Kill`, если возникает такая необходимость. Оператор с симпатичным названием `Kill` позволяет удалить файл с диска. В следующем коде делается попытка удаления всех файлов с таким же именем, как у сохраняемого файла, что позволит обойти предупреждение с запросом о перезаписи существующего файла.

Проблема заключается в том, что оператор `Kill` приводит к появлению фатальной ошибки, если удаляемый файл не существует. Если вызов оператора `Kill` привел к появлению ошибки, выполняется код обработки ошибки и оператор `Resume Next` передает управление следующему за оператором `Kill` оператору (`SaveAs`). Вызов функции `MsgBox` добавлен только в демонстрационных целях. В реальном приложении он обычно не используется:

```
Public Sub ErrorTrap2()
  Dim MyFile As String, Message As String
  Dim Answer As String

  On Error GoTo errorTrap

  Workbooks.Add
  MyFile = "C:\Data.xls"
  Kill MyFile
  ActiveWorkbook.SaveAs FileName:=MyFile
  ActiveWorkbook.Close

  Exit Sub

errorTrap:
  Message = "Номер ошибки: = " & Err.Number & vbCr
  Message = Message & Err.Description & vbCr & vbCr
  Message = Message & "Файл не существует"
  Answer = MsgBox(Message, vbInformation, "Ошибка")
  Resume Next
End Sub
```

Оператор On Error Resume Next

В качестве альтернативы оператору On Error GoTo можно использовать следующий оператор:

```
On Error Resume Next
```

Этот оператор приводит к игнорированию ошибок. Такая возможность должна применяться с осторожностью. Следующий код является результатом переработки процедуры ErrorTrap2:

```
Sub ErrorTrap3()
  Dim MyFile As String, Message As String

  Workbooks.Add
  MyFile = "C:\Data.xls"
  On Error Resume Next
  Kill MyFile
  On Error GoTo 0
  ActiveWorkbook.SaveAs FileName:=MyFile
  ActiveWorkbook.Close
End Sub
```

Оператор On Error Resume Next используется сразу перед оператором Kill. Если файл C:\Data.xls не существует, ошибка в работе оператора Kill игнорируется, и выполнение процедуры продолжается со следующей строки. В конце концов, нас не волнует отсутствие файла, так как именно этого мы пытаемся добиться с помощью оператора Kill.

Оператор On Error GoTo 0 может ввести в заблуждение. На самом деле его задачей является сброс объекта Err.

Оператор On Error Resume Next используется для создания кода, который в противном случае был бы менее эффективным. В следующей подпрограмме определяется существование имени в активной книге:

```
Public Sub TestForName()
  If NameExists("SalesData") Then
    MsgBox "Имя существует"
  Else
    MsgBox "Имя не существует"
```

```
  End If
End Sub

Public Function NameExists(myName As String) As Boolean
  Dim X As String
  On Error Resume Next
  X = Names(myName).RefersTo
  If Err.Number <> 0 Then
    NameExists = False
    Err.Clear
  Else
    NameExists = True
  End If
End Function
```

Процедура TestForName вызывает функцию NameExists, которая использует оператор On Error Resume Next для предотвращения фатальной ошибки при попытке назначения переменной имени из свойства RefersTo. В данном случае необходимости в операторе On Error GoTo 0 не возникает, так как обработка ошибок в процедуре отключается после завершения ее работы (хотя значение свойства Err.Number и не сбрасывается).

Если ошибка не возникает, свойство Number объекта Err остается равным нулю. Если свойство Err.Number имеет ненулевое значение, напрашивается вывод, что возникла ошибка. В этом случае можно предположить, что имя не существует, поэтому функция NameExists возвращает значение False и состояние ошибки сбрасывается. Альтернативой такой однопроходной функции является циклический перебор всех имен в книге в попытке найти совпадение. Если в книге присутствует большое количество имен, процесс может оказаться достаточно длительным.

Резюме

В этой главе были представлены компоненты языка VBA, которые позволяют создавать полезные и эффективные процедуры. Было показано, как добавлять в макросы интерактивность с помощью функций MsgBox и InputBox, а также как использовать переменные для хранения информации и получать справочную информацию о ключевых словах языка VBA.

Демонстрировалось, как объявлять переменные и определять их тип, а также влияние различных способов объявления на область видимости и время жизни переменных. В главе рассматривалось использование блочных структур If и Select Case для проверки условий и выполнения альтернативных вычислений. Циклы Do... Loop и For... Next позволяют эффективно повторять похожие вычисления. Также было рассказано, как использовать массивы, особенно в комбинации с циклическими процедурами. Кроме этого, было продемонстрировано использование операторов On Error для перехвата и обработки ошибок.

При создании кода VBA для Excel проще всего начинать с применения механизма записи макросов. После этого код можно модифицировать для адаптации к конкретным требованиям и повышения эффективности с помощью редактора VBE. Окна Object Browser (Просмотр объектов), Help (Справка) и справочный раздел в этой книге позволяют найти объекты, методы, свойства и события, которые не используются механизмом записи макросов. Язык VBA предоставляет структуры, позволяющие эффективно обрабатывать большие объемы данных и автоматизировать монотонные процессы.

Для программистов VBA редактор VBE является основным инструментом. В следующей главе будут рассмотрены некоторые возможности этого редактора. Знакомство с ним окажется полезным при рассмотрении более сложных тем программирования в главе 4.

2

Программирование в редакторе VBE

С момента появления Borland Turbo Pascal для DOS современные программисты получили предмет роскоши в виде интегрированной среды разработки, представляющей собой текстовый редактор (как Блокнот (Notepad) или Word), специально настроенный на написание кода на одном или нескольких языках программирования. Редактор VBE является интегрированной средой разработки, которая предоставляется вместе со всеми продуктами, поддерживающими язык VBA (включая Microsoft Excel). Таким образом, научившись использовать редактор VBE и язык VBA в одном приложении, можно считать, что умеешь это делать во всех приложениях. Остается только изучить объектную модель каждого приложения Office, для которого необходимо писать интересующий код. К счастью, ответы на большинство вопросов приводятся в справочном руководстве.

Эта книга посвящена программированию, поэтому желательно хорошо познакомиться с возможностями редактора VBE. Опытные программисты на языке VBA, которым уже доводилось использовать редактор VBE, могут просто просмотреть эту и следующую главы, чтобы убедиться, что в редакторе не появились новые, не встречавшиеся раньше средства. Новички в программировании на языке VBA могут почерпнуть из этой главы приемы оптимизации процессов написания и отладки кода.

Написание кода

Компьютеры ничего не знают о словах или правописании. Компьютер интересуется только битами и байтами. Даже языки низкого уровня с простым синтаксисом, например ассемблер, рассматриваются точно так же, как языки высокого уровня. Синтаксис и языки программирования интересуют только компилятор и человека. Так как задачей ком-

пилятора является преобразование языка программирования в биты и байты для выполнения кода компьютером, компилятор должен получать на входе код с правильным синтаксисом. Чем проще код для чтения человеком, тем проще его поддерживать. Следовательно, программист должен изучить синтаксис для обеспечения правильной работы компилятора и сформировать хороший стиль программирования для облегчения работы человека. Все эти принципы наиболее полно достигаются при использовании редактора.

Программирование для людей

Компилятор четко выполняет предоставленные инструкции; для этого код должен быть синтаксически правилен, а стиль программирования компилятор не интересует. Но кроме компилятора код пишется и для человека, поэтому стоит обратить внимание на несколько простых концепций и применять эти концепции в своей работе.

Обычно люди плохо воспринимают аббревиатуры, поэтому лучше использовать целые слова. Кроме этого, существительные и глаголы в английском языке составляют полное предложение, поэтому выбирайте хорошие существительные для имен классов и хорошие глаголы для имен методов. Более того, компилятор воспримет даже одну монолитную процедуру, в которой решается поставленная задача, пока синтаксис процедуры остается правильным. С другой стороны, люди испытывают значительные сложности в попытках понять смысл такого кода. Они не в состоянии одновременно воспринимать несколько идей и исключительно хорошо справляются с обработкой одной идеи в определенный момент. Поэтому методы и свойства должны быть короткими и однозначными, а не длинными и многозначными — другими словами, процедура должна решать одну задачу и имя процедуры отражать ее назначение.

Код намного сложнее понять, если в процедуре применяется больше нескольких строк кода. Если в процедуре необходимо использовать большое количество строк или очевидность кода под вопросом, напишите комментарий. Комментарии не всегда обязательны. Компилятор не обращает на них внимания, поэтому комментарии лучше использовать для прояснения алгоритма, а не для дублирования очевидного кода. Создание коротких процедур делает большие процедуры проще в отладке и позволяет разработчикам реорганизовывать существующие процедуры для решения новых задач. Часто монолитные процедуры могут использоваться только в одном контексте.

Кроме этого, оказывается полезной визуальная организация кода. Именно поэтому стоит использовать выравнивание для обозначения структуры. Старайтесь всегда применять одинаковые параметры выравнивания. Код является результатом труда программиста и должен быть правильно организован.

Старайтесь всегда явно и точно формулировать мысли. Используйте оператор `Option Explicit` в начале каждого модуля и явно объявляйте все переменные с помощью наиболее подходящего типа данных. Если значение не будет меняться в процессе работы, применяйте константу. Константы не меняются, поэтому если константа инициализирована с использованием подходящего значения, это значение всегда будет правильным и всегда будет приводить к получению надежного результата.

Наконец, тестируйте код небольшими фрагментами. Каждая процедура или свойство должны быть как можно более независимыми. Для этого нужно снижать зависимость от аргументов за пределами области видимости внутри процедуры. При создании кода процедуры (включая процедуры свойств) проверяйте каждую процедуру на получение ожидаемого результата при предоставлении известных параметров. Быстрое тестирование созданной процедуры можно выполнить в окне Immediate (Проверка). (Дополнительная информация о создании надежного кода, тестировании и отладке приводится в главах 7 и 8.)

Если следовать этим рекомендациям, получающийся в результате код будет правильным, единообразным и, несомненно, более простым для понимания. Результатом такого подхода к программированию будет большая готовность других программистов помочь в случае проблем в работе кода.

Написание кода в редакторе VBE

Чтение и написание большого объема кода — лучший способ получения опыта. Можно импортировать существующий код, который рассматривается далее в этой главе (дополнительная информация приводится в разделе “Импорт и экспорт кода Visual Basic”), но в большинстве случаев задачи программирования решаются с помощью редактора VBE.

Редактор VBE является отдельным приложением. Он связан с определенным инструментом Office. Для запуска редактора VBE в Excel (или в другом приложении Microsoft Office) можно нажать комбинацию клавиш <Alt+F11> или выбрать команду меню Сервис⇨Макрос⇨Visual Basic Editor (Tools⇨Macro⇨Visual Basic Editor). После запуска редактора VBE его можно воспринимать как специальный инструмент текстового процессора, поддерживающий синтаксис языка VBA. Вводимый текст будет распознаваться редактором VBE и редактор будет реагировать на синтаксические конструкции, состоящие из специальных ключевых слов. Например, редактор VBE будет автоматически конвертировать имена процедур в формат Pascal.

Помните, что все программисты имеют собственное мнение и склонность к определенному стилю написания кода, так как стиль кодирования — это совершенно субъективное понятие. Хорошим правилом является использование явно успешного стиля кодирования или использование стиля, который аргументировано более предпочтителен.

Куда делся мой код?

Весь код хранится в модулях, модулях классов или в объектах `UserForm`. Это специальные окна текстового процессора, используемые в редакторе VBE для обнаружения синтаксических ошибок и, с применением компилятора, программных ошибок. Модуль можно воспринимать, как пустой лист бумаги, а клавиатуру — как печатающую машинку. Если планируется написать короткий рассказ, можно сразу приступать к работе. Но большинство задач программирования требуют определенного планирования.

Существует несколько подходов к написанию кода. Опытные программисты начинают с использования инструмента проектирования, например, инструмента моделирования на языке Rational Unified Modeling Language (UML). Этот инструментарий поддерживает методический подход к решению проблем, позволяя описывать решение с помощью графических символов. Но изучение языка UML занимает некоторое время, так как это отдельный язык программирования. Язык моделирования обладает тем преимуществом, что графические символы лучше передают смысл и проще в создании и чтении, чем обычный код. Более простая форма инструмента моделирования позволяет рисовать диаграммы, которые описывают поток выполнения. Но инструмент для построения блок-схем предоставляет меньше возможностей, чем инструмент для моделирования UML, хотя и требует определенного времени на освоение. Кроме этого, существуют простые методы записи алгоритмов на бумаге обычным человеческим языком. Эта техника часто используется в курсах по компьютерным наукам в течение последних 20 лет. Особые программисты, которые называются архитекторами, должны знать все эти тонкости, но по ряду причин такой подход является не самой лучшей отправной точкой для начинающих программистов. Возможно, самым оптимальным началом для новичков является

написание кода, но код необходимо писать, помня об основном направлении. Рассмотрим более подробно значение этого утверждения.

Если необходимо добавить несколько простых макросов, можно начать с их записи. Это быстрый и простой подход. Если необходимо внести изменения, переключитесь в редактор VBE и модифицируйте записанные макросы. Если приходится писать приложение для Excel, можно комбинировать записанные макросы, собственные процедуры и диалоговые окна `UserForm`. После этого мысленно сгруппируйте проблемы и создайте отдельные модули, модули классов и диалоговые окна `UserForm` для каждой группы. Наконец, начинайте создавать поля, свойства, методы и события, позволяющие решить проблемы из каждой группы. Этот подход хорош для проблем средней сложности. Но для больших проблем подход "просто написать" может оказаться не самым понятным решением. На этом этапе справиться со сложным кодом поможет метод, который называется *рефакторингом (refactoring)*. Дополнительная информация о рефакторинге приводится в книге Мартина Фаулера (Martin Fowler) *Refactoring: Improving the Design of Existing Code* (Addison-Wesley, 1999). К сожалению, данная тема выходит за рамки нашей книги.

Последний совет: если заранее известно, что создается критическое приложение для Excel и у вас нет опыта в создании таких приложений, обратитесь за помощью как можно раньше. Учитель, архитектор или опытный помощник позволят обойти проблемы, которые могут возникнуть в процессе разработки. Рефакторинг является неотъемлемой частью уточнения, повторного использования и упрощения кода, но даже рефакторинг обладает определенными ограничениями. Критические приложения уровня предприятия должны проектироваться опытным архитектором или человеком, который уже имеет опыт успешного создания приложений сопоставимого уровня сложности. Лучше заранее осознать собственные ограничения и найти способ их компенсировать.

Управление проектом

Приложения VBA основаны на концепции проекта. Код в составе модулей является компонентом большего проекта, в который входят книги и листы. Из редактора VBE состав проекта можно определить с помощью окна Project Explorer (Окно проекта). Для открытия окна Project Explorer (Окно проекта) воспользуйтесь командой меню View⇨Project Explorer (Вид⇨Окно проекта) или нажмите комбинацию клавиш <Ctrl+R>. При создании новой книги окно Project Explorer (Окно проекта) должно выглядеть, как показано на рис. 2.1.

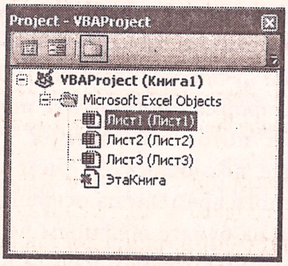


Рис. 2.1. Окно Project Explorer в новой книге

Лист1 является модулем класса, представляющим лист Лист1. Лист2 представляет лист Лист2, а Лист3 — лист Лист3. ЭтаКнига также будет модулем класса, который представляет книгу, открытую в данный момент. Эти модули являются представлением книг и листов с точки зрения VBA. Таким образом, при добавлении нового листа к принятой по умолчанию группе из трех листов в окне Project Explorer (Окно проекта) будет показан еще один модуль класса Лист4.

Хотя обычный пользователь может не догадываться о существовании модулей классов, представляющих каждый лист и книгу, разработчики должны об этом знать. Окно Project Explorer (Окно проекта) предоставляет средства для организации модулей и форм приложения. При необходимости здесь можно добавлять новые модули, модули классов и диалоговые окна UserForm. Эти сущности добавляются в проект с помощью меню Insert (Вставка) в редакторе VBE или с помощью контекстного меню в окне Project Explorer (Окно проекта). (Контекстные меню активизируются с помощью щелчка правой кнопкой мыши на объекте: в данном случае это щелчок правой кнопкой мыши на самом окне Project Explorer (Окно проекта).)

Управление расположением элементов управления

При добавлении в проект диалогового окна UserForm добавляется новый модуль формы (рис. 2.2). Кроме этого, открывается диалоговое окно Control Toolbox (Элементы управления). Добавление элементов управления в окно формы очень похоже на использование программы для рисования — щелкните на элементе управления в окне Control Toolbox (Элементы управления) и щелкните на том месте формы, в котором должен находиться элемент управления. Кроме этого, можно воспользоваться левой кнопкой мыши и перетащить прямоугольник, очерчивающий область, которую должен занимать элемент управления (см. рис. 2.2).

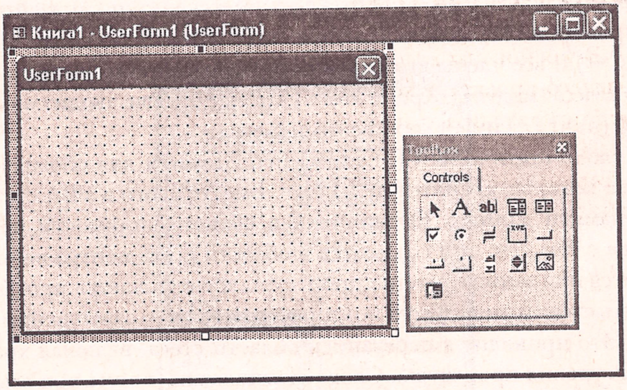


Рис. 2.2. Добавление диалогового окна UserForm

Точки на поверхности формы в процессе проектирования облегчают выравнивание элементов управления, помогая создавать визуально одинаковый интерфейс. Кроме этого, в меню редактора VBE Format (Формат) предоставляются несколько подменю, позволяющих управлять выравниванием, размером, центрированием, порядком, а также вертикальными и горизонтальными промежутками между элементами управления. Эти пункты меню работают с группами элементов управления. Для выделения нескольких

элементов управления необходимо щелкнуть левой кнопкой мыши и перетащить пунктирную линию вокруг интересующих элементов управления. (Также можно воспользоваться клавишей <Shift>. Удерживайте клавишу <Shift> и щелкайте на интересующих элементах управления, чтобы добавить их в группу.) Например, для выравнивания элементов управления `Label` по левому краю выделите интересующие элементы управления и выберите команду меню Format⇨Align⇨Left (Формат⇨Выравнивание⇨По левому краю). В большинстве случаев названия пунктов меню говорят сами за себя и, при наличии определенной практики, оказываются достаточно простыми и удобными.

Добавление классов

Модуль класса представляет собой особый тип модуля, в котором описывается класс Component Object Module (COM). Если речь заходит об объектно-ориентированном программировании вообще или объектно-ориентированном программировании в VBA, то подразумевается создание кода для модулей классов или кода, использующего существующие классы, например `Workbook`, `Worksheet` или `Range`.

Если определить модуль класса, то переменные этого типа (типа этого класса) придется определять с использованием имени модуля. Для создания и инициализации экземпляров класса применяются операторы `Set` и `New`. Простой модуль в составе Excel не требует использования этих операторов. Члены простых модулей технически являются статическими членами, то есть процедура в простом модуле — член единственного экземпляра класса (член модуля, в котором содержится процедура). Отвлечемся на минуту.

В языке C++ статические члены класса доступны до создания экземпляра класса (объекта); статические члены существуют в единственном экземпляре для всех экземпляров класса. В таких языках, как C++, программист должен явно определять все конструкции, поэтому использование статических членов в C++ требует дополнительных трудозатрат от программиста. Напротив, в таких языках, как VBA, простота достигается за счет снижения мощности доступных средств. Например, в VBA модуль является статическим классом (не с точки зрения VBA, а с точки зрения C++) и все члены класса доступны при указании имени модуля даже без создания экземпляра класса (модуля). Кроме этого (что еще больше вводит в заблуждение), в VBA слово статический (`static`) имеет собственный смысл.

В языке VBA слово статический (`static`) используется для обозначения переменных уровня процедуры, которые хранятся не в стеке. Вместо этого в VBA статические переменные хранятся в памяти данных. Ограниченная область видимости переменных уровня процедуры связана с тем, что они создаются в области стека центрального процессора, которая увеличивается при запуске каждой процедуры и уменьшается при завершении работы процедуры. Память стека используется повторно при каждом завершении одной процедуры и запуске другой. Это приводит к перезаписи области стека, включая хранящиеся там значения переменных уровня процедуры. Слово статический (`static`) в языке VBA означает, что переменная области процедуры хранится в памяти данных и не перезаписывается в результате уменьшения или увеличения области стека. Именно по этой причине статические переменные в VBA сохраняют значение между вызовами процедуры.

Если вы хотя бы раз страдали от насмешек программистов C++ над языком VBA, то должны уже понимать, что на самом деле для программирования на языке C++ (Delphi, VB.NET или C#) необходимо знать намного больше, чем для программирования на языке VBA (что иногда приводит к распуханию юношеского эго). Но в то же время стоит отметить, что существует множество проблем, которые быстрее и проще решаются с помо-

щью VBA. Кроме этого, на любом языке код могут писать как хорошие, так и плохие программисты. Если возникло устойчивое желание изучить такие концепции, как указатели, адреса, доопределение операторов, статические члены, интерфейсы, наследование, множественное наследование, многопоточность, кадры стека, полиморфизм, абстрактные классы, шаблоны и универсальные классы, изучите язык C++ и язык ассемблера. Если необходимо обрабатывать числа и данные, то эта книга именно то, что нужно.

Модификация свойств

В главе 1 было показано, что классы и модули имеют свойства. Это касается как новых, так и существующих классов. В редакторе VBE, как в инструменте визуального проектирования, могут модифицироваться классы специального вида, называемые элементами управления. Предоставляется возможность выделения классов элементов управления, а также изменения их состояния на этапе проектирования в окне Properties (Свойства).

Для открытия окна Properties (Свойства) в редакторе VBE необходимо выбрать команду меню View⇨Properties (Вид⇨Свойства) или нажать клавишу <F4>. В окне Properties (Свойства) предоставляется раскрывающийся список элементов управления, а также отсортированный в алфавитном порядке список открытых свойств. Свойства выбранного элемента управления можно модифицировать. Изменение значения свойства станет заметно уже на этапе проектирования. Например, для изменения заголовка объекта `UserForm1` из последнего раздела на `MyForm` достаточно щелкнуть на диалоговом окне `UserForm`, найти свойство `Caption` и изменить текст в поле ввода справа от имени свойства.

Некоторые свойства модифицируются так же легко, как свойство `Caption`. Другие свойства предоставляют более сложные механизмы модификации. Например, для изменения шрифта в диалоговом окне `UserForm` необходимо выделить диалоговое окно, открыть окно Properties (Свойства), найти свойство `Font` и щелкнуть на кнопке в поле ввода (рис. 2.3).

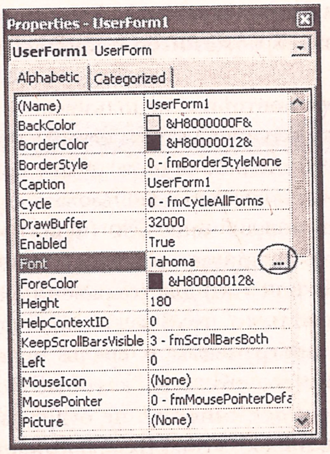


Рис. 2.3. Изменение шрифта в диалоговом окне UserForm

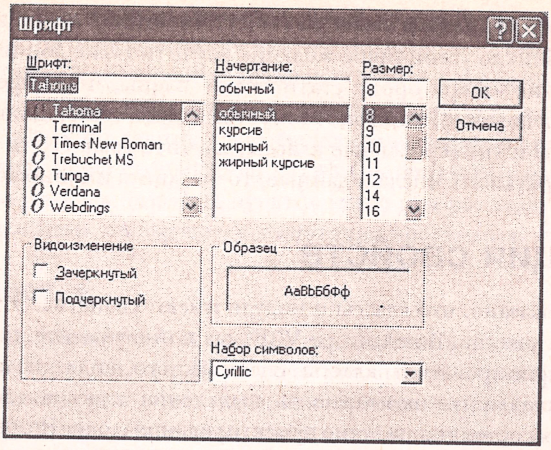


Рис. 2.4. Выбор шрифта для диалогового окна UserForm

Щелчок на кнопке приведет к запуску редактора этого свойства (см. рис. 2.4). После модификации свойства щелкните на кнопке OK. Внесенные изменения будут отражены в окне Properties (Свойства) и в элементе управления.

Импорт и экспорт кода Visual Basic

Код VBA и код Visual Basic практически ничем не отличаются. Это значит, что при необходимости поделиться кодом или при поиске подходящего кода можно искать как код, специально написанный для Microsoft Office, так и код, написанный для Visual Basic 6.0. То есть в мире существует несколько миллионов программистов на Visual Basic, с которыми можно объединить усилия. (Например, Пол Киммел ведет бесплатную газету для сайта `codeguru.com`, посвященную Visual Basic. Эта газета называется *Visual Basic Today*. Большая часть кода из этих статей может использоваться непосредственно при программировании на Excel VBA.)

Хорошим правилом является повторное использование как можно большего объема кода еще до написания собственного кода. Существующий код, скорее всего, уже был несколько раз вычитан и отлажен и, несомненно, намного лучше подходит для решения проблемы. Кроме чужого кода, не забывайте повторно использовать собственный код. Существующий код можно импортировать с помощью меню File (Файл) или контекстного меню в окне Project Explorer (Окно проекта). Например, выделите Sheet1 в Окне проекта и выберите Файл⇨Экспорт файла (File⇨Export File) для сохранения листа Sheet1 в файле `Sheet1.cls`. (Расширение `.cls` указывает на то, что Sheet1 является модулем класса.)

Помните, что различные инструменты и среды используют разные принятые по умолчанию параметры. Например, объекты, которые полностью доступны в составе проекта Workbook (`Workbook` и `Worksheet`), не будут автоматически предоставляться в Visual Basic 6. В этом случае в проект Visual Basic 6 придется добавлять ссылку на Microsoft Excel. Совместное использование кода Excel VBA между пользователями Excel не должно вызывать трудностей; совместное же использование кода с пользователями других версий Visual Basic может потребовать некоторых усилий. (Дополнительная информация о применении кода Visual Basic 6 в Excel приводится в главе 14.)

Редактирование

Выполнимость редактирования в среде VBE комбинирует в себе возможности текстового процессора и возможности, необходимые для работы с языком программирования. В меню Edit (Правка) предоставляются функции копирования, вставки и вырезания, а также поиска, замены текста и отмены операций. Кроме этого, команды меню Edit⇨Indent (Правка⇨Выровнять вправо) и Edit⇨Outdent (Правка⇨Выровнять влево) помогают при выравнивании блоков кода. Как и в других текстовых процессорах, в меню Edit (Правка) предоставляется возможность установки закладок в произвольных местах кода. Это позволит быстро перемещаться между интересующими фрагментами кода. Все доступные параметры предоставлены в меню Edit⇨Bookmark (Правка⇨Закладка).

Еще одной возможностью, характерной для редакторов кода, является автоматическое дополнение слов — Edit⇨Complete Word (Правка⇨Дополнить слово). Для вызова этой функции необходимо нажать комбинацию клавиш <Ctrl+пробел>, при этом появляется раскрывающееся меню со списком вариантов. Эта возможность не работает для ключевых слов. Например, если необходимо просмотреть список членов объекта `Worksheet`, просто введите имя объекта `Worksheet` и точку, после чего нажмите комбинацию клавиш <Ctrl+пробел>. Будет показан список всех членов класса `Worksheet`.

Современные объектные модели обычно оказываются слишком большими для запоминания. Лучшие программисты могут освоить синтаксис, распространенные идиомы и мощные шаблоны проектирования, а обязанность помнить особенности типов, а также порядок и количество параметров, перекладывается на справочное руководство и интеллектуальные редакторы.

Управление параметрами редактора

Изменение параметров редактора VBE требуется крайне редко. Самым значительным изменением может стать изменение ширины символа табуляции с 4 до 2 (рис. 2.5). Для просмотра и модификации параметров редактора выберите команду Tools⇨Options (Сервис⇨Параметры) в главном меню VBE.

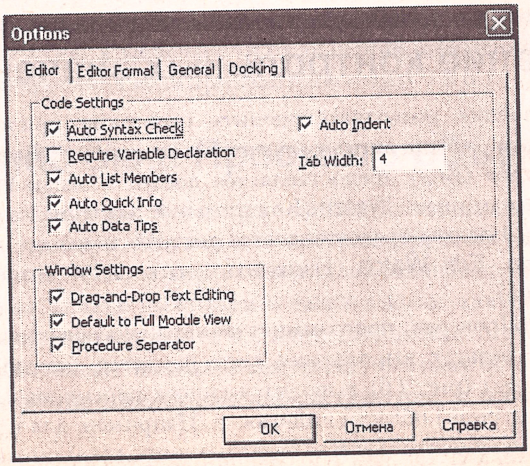


Рис. 2.5. Окно параметров редактора VBE

Опыт показывает, что самым распространенным изменением является увеличение размера шрифта при создании презентации. Принятый по умолчанию размер шрифта в 10 пикселей недостаточен при демонстрации презентации через проектор. В остальном в этом окне не предоставляются какие-то параметры, которые могут увеличить производительность труда, но знать о существовании такого окна все равно стоит.

Запуск и отладка кода

Знакомство с функциями записи и отладки кода в редакторе VBE позволяет значительно увеличить производительность труда. В меню Run (Выполнить) предоставляются команды Run Sub/UserForm (Выполнить подпрограмму/UserForm), Break (Прервать), Reset (Сбросить) и Design Mode (Режим конструктора). В меню Debug (Отладка) предоставляются команды Compile VBAProject (Откомпилировать проект VBA), Step Into (Шаг с заходом), Step Over (Шаг с обходом), Step Out (Шаг с выходом), Run To Cursor (Выполнить до текущей позиции), Add Watch (Добавить контрольное значение), Edit Watch (Изменить контрольное значение), Quick Watch (Контрольное значение), Toggle Breakpoint (Точка останова), Clear All Breakpoints (Снять все точки останова), Set Next Statement (Следующая инструкция) и Show Next Statement (Отобразить следующую инструкцию). Все эти пункты меню могут быть полезны, но чаще всего используются команды Run (Выполнить), Step Into (Шаг с заходом), Quick Watch (Контрольное значение) и Toggle Breakpoint (Точка останова).

Язык VBA является интерпретируемым, но поддерживается компиляция в PE-код (или переносимый исполнимый код). Пункты меню Run (Выполнить) и Step Into (Шаг с заходом) запускают процесс отладки. Пункт меню Quick Watch (Контрольное значение) открывает диалоговое окно, показывающее значение переменной, над именем которой находится курсор. Команда Toggle Breakpoint (Точка останова) вставляет низкоуровневое прерывание 3 и добавляет индикатор в виде красной точки напротив выбранной строки. Кроме этого, строка кода выделяется темно-красным цветом. Прерывание 3 является низкоуровневым прерыванием отладки, которое предоставляется BIOS компьютера. Использование точек останова позволяет запускать длительный процесс, останавливая выполнение как раз в тот момент, когда необходимо оценить состояние приложения. (Дополнительная информация об отладке и тестировании приводится в главах 7 и 8.)

Использование контрольных значений

Контрольные значения позволяют оценить состояние приложения. Окно Watch (Контрольные значения) можно активизировать из меню View (Вид) или из меню Debug (Отладка). В меню View (Вид) предоставляется доступ к окнам Immediate (Проверка), Locals (Локальные переменные), Watch (Контрольные значения) и Call Stack (Стек вызовов). В меню Debug (Отладка) предоставляется доступ к командам Add Watch (Добавить контрольное значение), Edit Watch (Изменить контрольное значение) и Quick Watch (Контрольное значение).

В окне Immediate (Проверка) программист может непосредственно вводить код и проверять его работу. Результат выполнения кода выдается немедленно. В окне Locals (Локальные переменные) (рис. 2.6) выводится список переменных, которые доступны в локальной (или процедурной) области видимости. Например, для класса `Worksheet` в окне Locals (Локальные переменные) всегда будет показана, как минимум, внутренняя ссылка на себя — объект `Me` (указатель на объект).

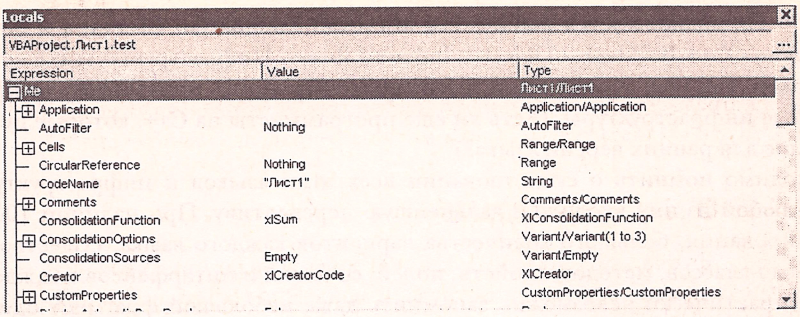


Рис. 2.6. Окно Локальные переменные

Окно Watch (Контрольные значения) ведет себя точно так же, как и окно Locals (Локальные переменные), но объекты в окно Watch (Контрольные значения) добавляются программистом. Окно Watch (Контрольные значения) позволяет следить за изменением состояния приложения в процессе выполнения. С другой стороны, в окне Quick Watch (Контрольное значение) предоставляется значение только одной переменной и эта информация выводится в модальном диалоговом окне; то есть, при использовании окна Quick Watch (Контрольное значение) работа приложения приостанавливается. Окна Locals (Локальные переменные), Watch (Контрольные значения) и Quick Watch (Контрольное значение) предоставляют информацию о состоянии объектов приложения.

В окне Call Stack (Стек вызовов) предоставляется обратный порядок вызовов функций в программе. Последний вызов указывается первым. После него указывается предпоследний вызов и т.д. Одним из применений окна Call Stack (Стек вызовов) является проверка порядка вызова методов. Кроме этого, окно обеспечивает быстрое перемещение между методами в окне редактора. Иногда окно Call Stack (Стек вызовов) незаменимо при отладке кода.

Окна Add Watch (Добавить контрольное значение) и Edit Watch (Изменить контрольное значение) являются модальными и используются для добавления или модификации значений в окне Watch (Контрольные значения). Контрольные значения могут быть простыми переменными, сложными объектами или действительными выражениями. Например, `A=10` является действительным контрольным значением, результатом которого будет `True` или `False`, в зависимости от значения переменной `A`. (Практические примеры отладки и тестирования кода приводятся в главах 7 и 8.)

Использование окна Просмотр объектов

Вероятно, существуют десятки объектных моделей для каждого языка программирования и каждой инфраструктуры. Мы неплохо программируем на таких языках, как C++, C, C#, Delphi, Visual Basic, VBA, и справляемся с ассемблером, Jscript, VBScript, Java, Clipper, Fox Pro, а также вполне выживаем в таких языках, как Cobol. Ключом к программированию на любом языке является изучение синтаксиса. К счастью, разные языки программирования часто имеют похожий синтаксис, поэтому остается только изучить объектную модель конкретного языка или инфраструктуры, а также принятые в языке конструкции и идиомы.

В Delphi используется собственная инфраструктура, которая называется Visual Control Library (VCL). В языках C# и VB.NET применяется инфраструктура .NET Framework.

В Microsoft C++ используется библиотека Microsoft Foundation Classes, а в Borland C++ (до появления C++ Builder) — Object Windows Library. Инфраструктуры существуют для таких языков, как Java. Кроме этого, некоторые сторонние производители предлагают собственные инфраструктуры. (Есть ли еще программисты на C++, которые помнят библиотеку Zinc для ранних версий языка?)

Необходимо помнить о существовании всех этих языков и инфраструктур, чтобы изучение любой из них входило в дальнейшую перспективу. При наличии 1000 языков программирования, большого количества вариантов каждого языка, существовании десятков тысяч классов, методов, свойств, полей, событий и интерфейсов в каждой инфраструктуре практически невозможно запомнить даже небольшой фрагмент одной из инфраструктур. Неплохой стратегией является изучение ключевых слов и синтаксиса каждого языка, который планируется использовать. После этого желательно ознакомиться с распространенными идиомами и конструкциями и в остальном полагаться на справочное руководство. Для этого в редакторе VBE (и множестве других инструментов) поддерживается окно Object Browser (Просмотр объектов). Выбрав команду View⇨Object Browser (Вид⇨Просмотр объектов) в меню редактора VBE, можно получить доступ к диалоговому окну Object Browser (Просмотр объектов). В этом окне предоставляется список классов и их членов в иерархически структурированной форме. Дополнительная информация об интересующих объектах приводится в справочном руководстве. Таким образом, со временем сформируются устойчивые знания о тех частях инфраструктуры, которые используются постоянно.

Начинающим разработчикам можно посоветовать найти решение в составе объектной модели VBA еще до того, как оно будет создаваться с нуля. Если интересующий результат отсутствует в объектной модели VBA или в справочном руководстве, обратитесь к Windows API (дополнительная информация приводится в главе 16). Но даже если это не принесло результата, перед созданием собственного решения поищите готовое решение в сети Internet или среди предложений сторонних производителей. Наконец, создавайте решение только в том случае, если это действительно необходимо. Даже пара часов поиска существующего решения оказывается более полезным использованием рабочего времени, чем реализация тривиальных алгоритмов с нуля.

Резюме

Профессиональный рост в программировании требует постоянной практики. Как и любая умственная или физическая деятельность, программирование заставляет учиться мозг и тело. В этой главе были перечислены некоторые ключевые возможности редактора VBE, которые будут использоваться на протяжении всей книги. Почему эти возможности рассматриваются здесь? Недавно мы работали с группой разработчиков, которые программировали в течение года и не знали о существовании окна Quick Watch (Контрольное значение). Теперь вы знаете.

Редактор VBE является мощным инструментом, предназначенным для работы с языком программирования Visual Basic for Applications. Рекомендуем потратить некоторое время на ознакомление с главным меню редактора и контекстными меню, а также панелями инструментов каждого компонента редактора. Скорее всего, в процессе ознакомления будут найдены возможности, которые не использовались ранее. Это позволит работать более эффективно как с этой книгой, так и в процессе программирования.

3

Объект Application

В этой главе рассматривается подмножество функциональности Excel. Здесь описываются возможности, которые не обязательно связаны друг с другом. В общем, объектная модель Excel содержит объекты, предназначенные для решения вполне определенных задач. Объект Application находится на вершине объектной модели Excel и содержит все остальные объекты. Кроме этого, объект Application выступает хранилищем для свойств и методов, которые не подходят для включения в любой другой объект, но необходимы для программного управления Excel. Например, существуют свойства объекта Application, предназначенные для управления обновлением экрана и включения предупреждений. Существует метод объекта Application, подсчитывающий формулы в открытых книгах.

Глобальные члены

Многие методы и свойства объекта Application являются членами группы <globals>, доступной в самом начале списка классов в окне Object Browser (Просмотр объектов) (рис. 3.1).

Если свойство или метод входит в группу <globals>, на него можно ссылаться, не указывая ссылку на объект. Например следующие две ссылки эквивалентны:

```
Application.ActiveCell
ActiveCell
```

Но следует соблюдать осторожность. Некоторые часто используемые свойства объекта Application, например ScreenUpdating, не являются глобальными. Показанный ниже код является корректным:

```
Application.ScreenUpdating = False
```

А результат использования следующего кода может оказаться неожиданным:

```
ScreenUpdating = False
```

Этот код создает переменную и присваивает ей значение `False`. Для избежания подобных ошибок стоит добавить строку `Option Explicit` в начале каждого модуля. После этого на этапе компиляции все ссылки подобного рода будут помечены как неопределенные переменные.

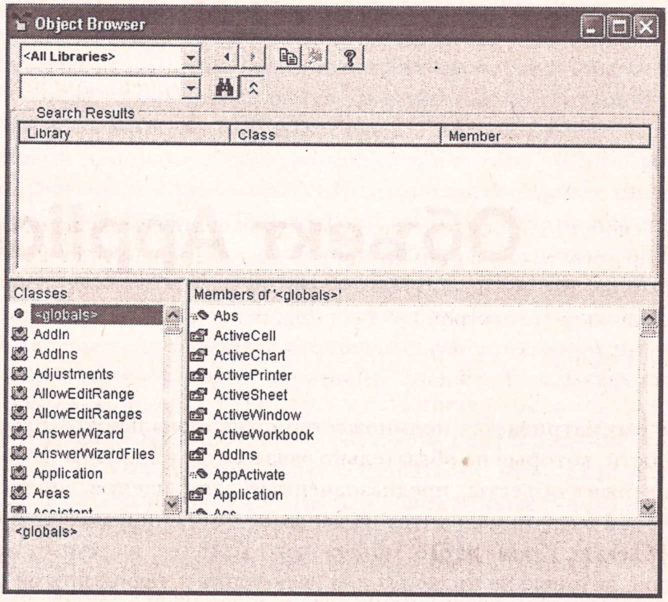


Рис. 3.1. Окно Просмотр объектов. Группа <globals>

> Помните, что оператор `Option Explicit` может добавляться автоматически при создании нового модуля. Для этого выберите команду меню Tools⇨Options (Сервис⇨Параметры) в редакторе VBE и на вкладке Editor (Редактор) установите флажок Требовать объявления переменных (Require Variable Declaration).

Свойства Active

Объект `Application` предоставляет множество ссылок, которые можно применять для обращения к активным объектам без указания явного имени. Это позволяет использовать объекты, активные в момент выполнения макроса. Кроме этого, эти свойства дают возможность создавать универсальный код, который работает с объектами одного и того же типа, но имеющими разные имена.

Следующие свойства объекта `Application` являются глобальными и позволяют ссылаться на активные объекты.

- `ActiveCell`
- `ActiveChart`
- `ActivePrinter`

- ActiveSheet
- ActiveWindow
- ActiveWorkbook
- Selection

Если только что была создана новая книга и ее необходимо сохранить под конкретным именем, воспользуйтесь свойством ActiveWorkbook для получения ссылки на объект Workbook.

```
Workbook.Add
ActiveWorkbook.SaveAs Filename:="C:\Data.xls"
```

Если необходимо создать макрос, включающий полужирный шрифт для выделенных ячеек, можно воспользоваться свойством Selection. Оно позволяет получить ссылку на объект Range, в который входят выделенные ячейки.

```
Selection.Font.Bold = True
```

Помните, что свойство Selection не будет возвращать ссылку на объект Range, если выделен объект другого типа, например Shape, или активный лист не является листом электронной таблицы. Возможно, в макрос потребуется добавить условие, которое будет проверять, выделен ли лист электронной таблицы, перед тем как вставлять данные.

```
If TypeName(ActiveSheet) <> "Worksheet" Or _
   TypeName(Selection) <> "Range" Then _
      MsgBox "Этот макрос может использоваться только вместе с
↳ диапазоном", vbCritical
   Exit Sub
End If
```

Вывод предупреждений

Реагирование на предупреждения приложения во время работы макроса может оказаться утомительным. Например, если макрос удаляет лист, выдается окно предупреждения с кнопкой OK и Отмена (Cancel). Пользователь может щелкнуть на кнопке Отмена (Cancel), что приведет к отмене операции удаления и отрицательно повлияет на работу остального кода макроса, в котором делается предположение об успешном завершении удаления.

Большинство предупреждений можно отключить, установив свойство DisplayAlerts в значение False. При подавлении диалоговых окон с предупреждениями автоматически выполняется операция, связанная с принятой по умолчанию кнопкой диалогового окна, например:

```
Application.DisplayAlerts = False
ActiveSheet.Delete
Application.DisplayAlerts = True
```

> Устанавливать свойство DisplayAlerts в значение True по завершении работы макроса необязательно, так как интерпретатор VBA устанавливает свойство автоматически. Но обычно имеет смысл устанавливать свойство сразу после подавления определенного сообщения, чтобы неожиданные предупреждения выдавались на экран.

Свойство DisplayAlerts обычно используется для подавления предупреждений о перезаписи существующих файлов с помощью команды Файл⇨Сохранить Как (File⇨SaveAs). При подавлении этого предупреждения выполняется действие, принятое по умолчанию, и файл перезаписывается без прерывания работы макроса.

Обновление экрана

Мерцание экрана во время работы макроса может раздражать. Это происходит при выделении или активизации объектов и обычно характерно для макросов, сгенерированных автоматически.

> Стоит избегать выделения объектов средствами кода VBA. Это редко когда требуется, и при отказе от выделения или активизации объектов код будет работать быстрее. Большая часть кода в этой книге работает без выделения объектов.

Если экран необходимо зафиксировать на время работы макроса, воспользуйтесь следующим кодом:

```
Application.ScreenUpdating = False
```

Экран будет заморожен до установки свойства `ScreenUpdating` в значение `True` или до завершения работы макроса и передачи управления пользовательскому интерфейсу. Присваивать переменной `ScreenUpdating` значение `True` стоит только в том случае, если экран должен обновиться еще в процессе работы макроса.

Существует одна ситуация, когда желательно установить свойство `ScreenUpdating` в значение `True` еще до завершения работы макроса. Если в процессе работы макроса на экран выводится диалоговое окно `UserForm` или встроенное диалоговое окно, обновление экрана стоит включить еще до вывода объекта на экран. Если пользователь перетащит диалоговое окно `UserForm` в то время, когда обновление экрана отключено, контур окна будет оставаться по всему пути окна, перекрывая содержимое экрана. Обновление экрана можно отключить после вывода объекта на экран.

> Положительным побочным эффектом отключения обновления экрана является ускорение работы кода. Ускоряется даже код, который не использует выделение объектов и не нуждается в обновлении экрана. Для получения максимальной производительности откажитесь от выделения объектов и отключите обновление экрана.

Оценка

Метод `Evaluate` может использоваться для расчета значения формул листов Excel и генерации ссылок на объекты `Range`. Стандартный синтаксис вызова метода `Evaluate` выглядит следующим образом:

```
Evaluate("Выражение")
```

Кроме этого, существует сокращенная форма вызова, в которой отсутствуют двойные кавычки, а выражение заключается в квадратные скобки, например:

```
[Выражение]
```

На месте `Выражения` может находиться любое действительное выражение на листе с или без знака равенства слева. Также это может быть ссылка на диапазон ячеек. Расчеты на листе могут включать в себя функции, недоступные в VBA через объект `WorksheetFunction`. Также это могут быть формулы массивов на листе. Дополнительная информация об объекте `WorksheetFunction` приводится далее в этой главе.

Например, функция `ISBLANK`, которую можно использовать в собственных формулах на листе, недоступна для кода VBA через объект `WorksheetFunction`, так как эквива-

лентная функция VBA IsEmpty предоставляет те же возможности. Хотя при желании можно использовать и функцию ISBLANK. Следующие два примера являются эквивалентными и возвращают значение True, если ячейка A1 пустая, и False в противном случае:

```
MsgBox Evaluate("=ISBLANK(A1)")
MsgBox [ISBLANK(A1)]
```

Преимуществом первого подхода является возможность генерации строкового значения в результате работы кода. Это делает код исключительно гибким. Второй прием имеет более короткую запись, но выражение можно изменить только с помощью редактирования кода. Следующая процедура выводит значение True или False в зависимости от пустоты активной ячейки. Кроме этого, в процедуре демонстрируется гибкость первого подхода:

```
Public Sub IsActiveCellEmpty()
  Dim FunctionName As String, CellReference As String
  FunctionName = "ISBLANK"
  CellReference = ActiveCell.Address
  MsgBox Evaluate(FunctionName & "(" & CellReference & ")")
End Sub
```

Обратите внимание, что второй прием не позволяет оценивать значение выражения, содержащего переменные.

В следующем примере показаны два способа использования метода Evaluate для генерации ссылки на объект Range с присвоением значения этому объекту:

```
Evaluate("A1").Value = 10
[A1].Value = 10
```

Эти выражения эквивалентны. Выражение можно сократить еще больше, опустив свойство Value, так как это принятое по умолчанию свойство объекта Range:

```
[A1] = 10
```

Более интересным примером использования метода Evaluate является возврат содержимого коллекции имен книг Names с эффективной генерацией массивов значений. В следующем коде создается скрытое имя для хранения пароля. Скрытые имена не видны в диалоговом окне Insert⇨Name⇨Define (Вставка⇨Имя⇨Определение), поэтому они являются удобным методом хранения информации о книге с сохранением чистоты пользовательского интерфейса.

```
Names.Add Name:="PassWord", RefersTo:="Bazonkas", Visible:=False
```

После этого скрытые данные можно использовать в следующих выражениях:

```
UserInput = InputBox ("Введите пароль")
If UserInput = [PassWord] Then
...
```

Применение имен для хранения данных рассматривается в главе 21.

Кроме этого, метод Evaluate можно использовать вместе с массивами. В следующем выражении генерируется двумерный массив типа Variant на 100 строк и один столбец. В массиве хранятся значения от 101 до 200. Этот код работает эффективнее, чем цикл For... Next:

```
RowArray = [ROW(101:200)]
```

Точно так же следующий код присваивает значения от 101 до 200 диапазону B1:B100. И в этом случае используется меньший объем кода, чем в цикле `For... Next`:

```
[B1:B100] = [ROW(101:200)]
```

Метод InputBox

В языке VBA существует функция `InputBox`, которая предоставляет простой способ получения данных у пользователя. Кроме этого, метод `InputBox` предоставляется объектом `Application`. Метод объекта `Application` предоставляет похожее диалоговое окно, но имеет больше возможностей. Этот метод позволяет управлять типом данных, которые может вводить пользователь, а также обнаруживать нажатие клавиши Отмена (Cancel).

Если в окне используется неквалифицированный вызов функции `InputBox` (как показано ниже), будет вызвана функция `InputBox` из языка VBA:

```
Answer = InputBox(prompt:="Введите диапазон")
```

Пользователь может вводить в диалоговое окно только данные. Выбор ячейки с помощью указателя мыши невозможен. Возвращаемое значение функции `InputBox` всегда имеет строковый тип и проверка содержимого строки никогда не выполняется. Если пользователь ничего не вводит, возвращается строка нулевой длины. Если пользователь щелкает на кнопке Отмена (Cancel), также возвращается строка нулевой длины. Код не позволяет различить отсутствие ввода и щелчок на кнопке Отмена (Cancel).

В следующем примере применяется метод `InputBox` объекта `Application`. В открывшемся диалоговом окне выдается запрос на выбор диапазона:

```
Answer = Application.InputBox(prompt:="Введите диапазон", Type:=8)
```

Параметр `Type` может принимать следующие значения (или сумму этих значений):

Значение	*Описание*
0	Формула
1	Число
2	Текст (строка)
4	Логическое значение (`True` или `False`)
8	Ссылка на ячейку, как объект `Range`
16	Значение ошибки, например, `#N/A`
64	Массив значений

Пользователь может выбирать ячейки с помощью мыши или вводить данные. Если введены данные неподходящего типа, метод `InputBox` выдает сообщение об ошибке и запрашивает ввод новых данных. Если пользователь щелкает на кнопке Отмена (Cancel), метод `InputBox` возвращает значение `False`.

Если возвращаемое значение присваивать переменной типа `Variant`, для большинства типов данных можно сравнить возвращаемое значение с `False`, чтобы определить щелчок на кнопке Отмена (Cancel). Если запрашивается диапазон, то возникает более сложная ситуация. В этом случае необходимо использовать следующий код:

```
Public Sub SelectRange()
  Dim aRange As Range

  On Error Resume Next
  Set aRange = Application.InputBox(prompt:="Введите диапазон",
↳ Type:=8)
  If aRange Is Nothing Then
    MsgBox "Операция отменена"
  Else
    aRange.Select
  End If
End Sub
```

Во время выполнения этого кода вывод будет выглядеть, как показано на рис. 3.2.

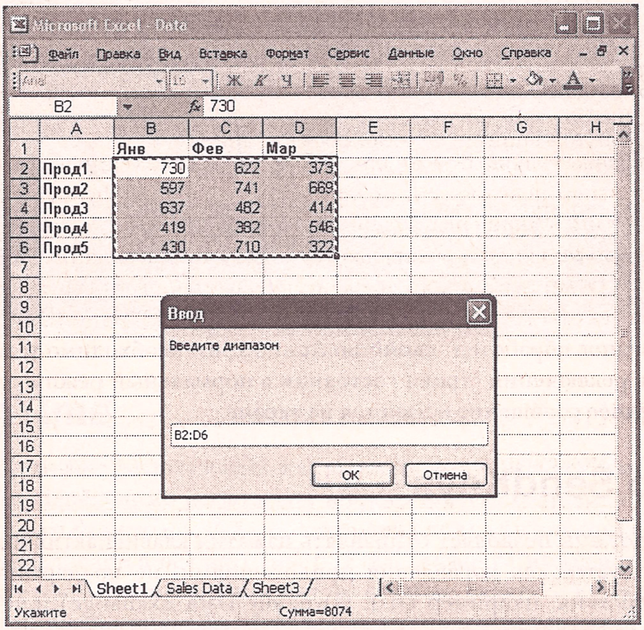


Рис. 3.2. Диалоговое окно, предоставляемое методом Application.InputBox

Проблема заключается в том, что для назначения объекта `Range`, объектной переменной необходимо использовать оператор `Set`. Если пользователь щелкает на кнопке Отмена (Cancel) и возвращается значение `False`, оператор `Set` неудачно завершает работу, и возникает ошибка времени выполнения. Использование оператора `On Error Resume Next` позволяет избавиться от ошибки времени выполнения и проверить генерацию правильного диапазона. Известно, что встроенные механизмы проверки типов метода `InputBox` проверяют действительность диапазона при щелчке на кнопке OK. Если метод возвращает пустой диапазон, то можно предположить, что пользователь щелкнул на кнопке Отмена (Cancel).

Строка состояния

Свойство StatusBar позволяет управлять текстом, который отображается слева в строке состояния Excel в нижней части экрана. Строка состояния предоставляет средство информирования пользователей во время работы большого макроса. Пользователь должен знать, что программа работает, особенно если отключено обновление экрана и приложение не проявляет никакой визуальной активности. Даже если обновление экрана отключено, в строке состояния можно выводить сообщения.

Этот прием может применяться в процедуре с циклом (следующий код демонстрирует такое использование строки состояния):

```
Public Sub ShowStatus()

  Application.ScreenUpdating = False

  Dim I As Long
  For I = 0 To 10000000
    If I Mod 1000000 = 0 Then
      Application.StatusBar = "Обработка записи" & I
    End If
  Next I
  Application.StatusBar = False

  Application.ScreenUpdating = True

End Sub
```

После завершении обработки свойство StatusBar необходимо установить в значение False для переключения строки состояния в нормальный режим работы. В противном случае последнее сообщение останется на экране.

Свойство SendKeys

Свойство SendKeys позволяет отправлять нажатия клавиш активному окну. Эта возможность используется для управления приложениями, не поддерживающими другие формы взаимодействия, например DDE (Dynamic Data Exchange) или OLE, и обычно рассматривается как последнее средство.

В следующем примере открывается приложение Блокнот (Notepad), которое не поддерживает DDE и OLE. После открытия в документ записывается строка данных:

```
Public Sub SendKeyTest()
  Dim ReturnValue As Double
  ReturnValue = Shell("NOTEPAD.EXE", vbNormalFocus)
  Call AppActivate(ReturnValue)

  Application.SendKeys "Copy Data.xls c:\", True
  Application.SendKeys " ", True
  Application.SendKeys "%FABATCH ", True
End Sub
```

> Важно понять, что функция SendKeys отправляет нажатия клавиш активному приложению. Если этот макрос запустить в редакторе VBE, редактор получит фокус после вызова функции SendKeys и нажатые клавиши будут отправлены редактору. Для правильного сохранения фокуса на приложении Блокнот (Notepad) запускайте этот макрос из Excel.

Процедура SendKeyTest использует комбинацию клавиш <Alt+F+A> для выполнения операции Файл⇨Сохранить Как (File⇨SaveAs) и вводит имя файла BATCH. Символ % используется вместо клавиши <Alt>, а символ ~ — вместо клавиши <Enter>. Символ ^ используется вместо клавиши <Ctrl>. Для представления других специальных клавиш их имена указываются в фигурных скобках, например, для передачи нажатия клавиши <Del> применяется последовательность символов {Del} (передача нажатий этих клавиш показана в следующем примере). Нажатия клавиш можно отправлять и в Excel. В следующей процедуре очищается содержимое окна Immediate (Проверка). Если окно Immediate (Проверка) использовалось для экспериментирования или вывода отладочной информации с помощью вызова Debug.Print, в этом окне может содержаться большой объем лишней информации. Эта процедура переключает фокус в окно Immediate (Проверка) и отправляет нажатие комбинации клавиш <Ctrl+a> для выделения всего текста в окне. После этого текст удаляется с помощью нажатия клавиши <Del>.

```
Public Sub ImmediateWindowClear()
       Application.VBE.Windows.Item("Immediate").SetFocus
       Applicatoin.SendKeys "^a"
       Application.SendKeys "{Del}"
End Sub
```

> Для правильной работы этого макроса должен быть разрешен программный доступ к проекту Visual Basic. Для разрешения доступа выберите в меню Excel (не VBE) команду Сервис⇨Макросы⇨Безопасность (Tools⇨Macros⇨Security) и активизируйте вкладку Надежные источники (Trusted Sources).

Метод OnTime

Метод OnTime можно использовать для планирования запуска макроса на какой-то момент в будущем. Необходимо указать дату и время запуска макроса, а также имя макроса. Если воспользоваться методом Wait объекта Application для приостановки работы макроса, приостанавливается вся активность Excel, включая интерактивное взаимодействие с пользователем. Преимуществом метода OnTime является возможность возврата к нормальному взаимодействию с Excel, включая запуск других макросов в процессе ожидания запланированного запуска.

Предположим, существует открытая книга со ссылками на книгу Data.xls, которая хранится на сетевом сервере, но еще не открыта. В 15:00 необходимо обновить связи с книгой Data.xls. Следующий пример планирует запуск макроса RefreshData в 15:00 в этот же день. Функция Date возвращает текущую дату, а функция TimeSerial используется для добавления необходимого времени:

```
Public Sub RunOnTime()
       Application.OnTime Date + TimeSerial(15, 0, 0), "RefreshData"
End Sub
```

> Стоит обратить внимание, что при попытке запуска макроса после 15:00 будет выдано сообщение об ошибке, так как планирование запуска на момент в прошлом невозможно. При необходимости можно изменить время запуска, чтобы момент запуска находился в будущем.

В следующем макросе RefreshData связи с книгой Data.xls, находящиеся в книге ThisWorkbook, обновляются с помощью метода UpdateLink. Объект ThisWorkbook является удобным способом ссылки на книгу, в которой находится макрос:

```
Public Sub RefreshData()
      ThisWorkbook.UpdateLink Name:="c:\Data.xls", Type:=xlExcelLinks
End Sub
```

Если данные необходимо обновлять регулярно, макрос можно заставить запускать самого себя:

```
Dim ScheduledTime As Date

Public Sub RefreshData()
  ThisWorkbook.UpdateLink Name:="C:\Data.xls", Type:=xlExcelLinks
  ScheduledTime = Now + TimeSerial(0, 1, 0)
  Application.OnTime ScheduledTime, "RefreshData"
End Sub

Public Sub StopRefresh()
  Application.OnTime ScheduledTime, "RefreshData", , False
End Sub
```

После запуска подпрограммы RefreshData она будет планировать собственный запуск каждую минуту. Для остановки макроса необходимо знать время запланированного запуска, поэтому переменная уровня модуля ScheduledTime содержит время следующего запланированного запуска. Подпрограмма StopRefresh устанавливает четвертый параметр метода OnTime в значение False, что приводит к отмене запланированного запуска подпрограммы RefreshData.

> При планировании запуска макроса в будущем с помощью метода OnTime необходимо обеспечить присутствие Excel в памяти до запланированного момента времени. При этом книга, содержащая макрос, не обязательно должна быть открыта. При необходимости Excel откроет книгу автоматически.

Кроме этого, метод OnTime может оказаться полезным для приостановки обработки до возникновения неподконтрольного события. Например, с помощью DDE данные можно отправить другому приложению, но до продолжения обработки необходимо подождать ответа от получателя данных. Для этого создается два макроса. Первый отправляет данные и планирует запуск второго, который обрабатывает ответ через определенный промежуток времени. Второй макрос может продолжать работу, пока на листе или в окружении не будут обнаружены изменения, вызванные ответом внешнего приложения.

Метод OnKey

Метод OnKey может использоваться для назначения макроса нажатию клавиши или нажатию любой комбинации клавиш <Ctrl>, <Shift> или <Alt> с любой другой клавишей. Этот метод также применяется для отключения определенных комбинаций клавиш.

В следующем примере показано, как назначить макрос DownTen клавише управления курсором <вниз>. После запуска макроса AssignDown нажатие клавиши <вниз> будет приводить к запуску макроса DownTen и перемещению выделения на 10 строк вместо одной:

```
Public Sub AssignDown()
  Application.OnKey "{Down}", "DownTen"
End Sub
```

```
Public Sub DownTen()
  ActiveCell.Offset(10, 0).Select
End Sub

Public Sub ClearDown()
  Application.OnKey "{Down}"
End Sub
```

Метод ClearDown возвращает клавише <вниз> стандартную функциональность.

Метод OnKey может использоваться для отключения существующих комбинаций клавиш. Например, для отключения комбинации клавиш <Ctrl+c>, обычно используемой для копирования, можно применить следующий код. Этот код присваивает комбинации клавиш пустую процедуру:

```
Sub StopCopyShortCut()
    Application.OnKey "^c", ""
End Sub
```

Обратите внимание, что используется обозначение "c" в нижнем регистре. Верхний регистр ("C") будет описывать комбинацию клавиш <Ctrl+Shift+c>. Для восстановления функциональности комбинации клавиш <Ctrl+c> воспользуйтесь следующей процедурой:

```
Sub ClearCopyShortCut()
    Application.OnKey "^c"
End Sub
```

> Назначения, сделанные с помощью метода OnKey, относятся ко всем открытым книгам и сохраняются в течение всего сеанса работы в Excel.

Функции листа

В коде Excel VBA можно использовать код из двух источников. Одна группа функций является частью языка VBA, а вторая — подмножеством функций листов Excel.

Excel и язык Visual Basic (в виде VBA) не были одним целым до выхода Excel 5. Каждая система разрабатывалась отдельно и имела собственные функции. Это привело к существованию нескольких пересечений и конфликтов между множествами функций. Например, в Excel есть функция DATE, а в VBA — функция Date. Функция Excel DATE принимает три аргумента (год, месяц и день) для генерации конкретной даты. Функция VBA Date не имеет аргументов в возвращает текущую дату, полученную от системных часов. Кроме этого, в VBA существует функция DateSerial, которая принимает те же аргументы и возвращает тот же результат, что и функция Excel DATE. Наконец, функция Excel TODAY не имеет аргументов и возвращает тот же результат, что и функция VBA Date.

Как правило, если функция VBA выполняет ту же задачу, что и функция Excel, функция Excel оказывается недоступна для кода VBA (хотя метод Evaluate позволяет получить доступ к любой функции Excel, как было показано в предыдущей главе). Кроме этого, стоит особо отметить функцию Excel MOD, которая недоступна непосредственно для кода VBA, но оператор VBA Mod решает ту же задачу. В следующей строке кода используется метод Evaluate для вывода номера дня недели с помощью функции Excel MOD и функции Excel TODAY:

```
MsgBox [MOD(TODAY(),7]
```

Тот же результат можно получить при использовании функции Date и оператора Mod:

```
MsgBox Date Mod 7
```

Кроме этого, в VBA недоступна функция CONCATENATE. Вместо этого можно воспользоваться оператором & (который можно использовать и в формуле на листе Excel). Если необходимо применить функцию CONCATENATE в коде VBA, создайте следующий код:

```
Public Sub ConcatenateExample1()
   Dim X As String, Y As String
   X = "Jack "
   Y = "Smith"
   MsgBox Evaluate("CONCATENATE(""" & X & """,""" & Y & """)")
End Sub
```

С другой стороны, можно отказаться от лишней работы и получить тот же результат с использованием следующего кода:

```
Public Sub ConcatenateExample2()
   Dim X As String, Y As String
   X = "Jack "
   Y = "Smith"
   MsgBox X & Y
End Sub
```

Такие функции VBA, как `Date`, `DateSerial` и `IsEmpty`, могут применяться без квалификации, так как они входят в группу `<globals>`. Например, следующий код вполне допустим:

```
StartDate = DateSerial(2003, 9, 6)
```

Такие функции Excel, как `VLOOKUP` и `SUM`, являются методами объекта `WorksheetFunctions` и требуют использования следующего синтаксиса:

```
Total = WorksheetFunctions.Sum(Range("A1:A10"))
```

Для совместимости с Excel 5 и Excel 95 вместо объекта `WorksheetFunctions` можно использовать объект `Application`:

```
Total = Application.Sum(Range("A1:A10"))
```

> В некоторых версиях VBA вызов `WorksheetFunctions.Vlookup` работает неправильно. Гарантированная работоспособность обеспечивается вызовом `Application.Vlookup`.

Полный список функций листов, непосредственно доступных в VBA, предоставляется в описании объекта `WorksheetFunctions` в соответствующем разделе справочного руководства.

Свойство Caller

Это свойство объекта `Application` возвращает ссылку на объект, который вызвал или запустил макрос. Оно широко применялось в Excel 5 и Excel 95, где использовалось вместе с меню и элементами управления на диалоговых листах. В Excel 97 и более новых версиях панели инструментов и элементы управления ActiveX в диалоговых окнах `UserForm` пришли на замену меню и элементам управления на диалоговых листах. В новых возможностях Excel свойство `Caller` не используется.

Свойство `Caller` все еще применяется совместно с элементами управления с панели инструментов **Формы** (Forms), с объектами рисования, к которым подключены макросы, и с определенными пользователем функциями. В частности, это свойство позволяет найти

ячейку, из которой была вызвана определенная пользователем функция. На показанном на рис. 3.3 функция `WorksheetName` используется для вывода имени листа в ячейке B2.

При применении в функции свойство `Application.Caller` возвращает ссылку на ячейку, из которой была вызвана эта функция. Свойство возвращает ссылку в виде объекта `Range`. В следующей функции `WorksheetName` свойство `Parent` объекта `Range` используется для генерации ссылки на объект `Worksheet`, в котором содержится объект `Range`. Свойству `Name` объекта `Worksheet` присваивается возвращаемое значение функции. Метод `Volatile` объекта `Application` заставляет Excel пересчитывать функцию при каждом пересчете содержимого листа, поэтому при изменении имени функция покажет новое имя листа.

```
Public Function WorksheetName() As String
   Application.Volatile
   WorksheetName = Application.Caller.Parent.Name
End Function
```

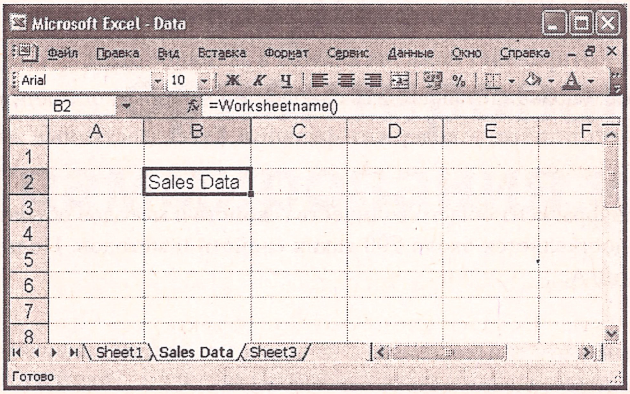


Рис. 3.3. Использование свойства Caller

Использование следующего кода в функции `WorksheetName` можно считать ошибкой:

```
WorksheetName = ActiveSheet.Name
```

Если пересчет выполняется в то время, когда активен другой лист, функция вернет неправильное имя листа.

Резюме

В этой главе были описаны некоторые полезные свойства и методы объекта `Application`. Так как объект `Application` используется для предоставления универсальной функциональности, которая не подпадает под назначение других объектов, некоторые из очень полезных возможностей можно пропустить.

В этой главе были рассмотрены следующие методы и свойства:

- ❑ `ActiveCell` — содержит ссылку на активную ячейку;
- ❑ `ActiveChart` — содержит ссылку на активную диаграмму;
- ❑ `ActivePrinter` — содержит ссылку на активный принтер;
- ❑ `ActiveSheet` — содержит ссылку на активный лист;

- ActiveWindow — содержит ссылку на активное окно;
- ActiveWorkbook — содержит ссылку на активную книгу;
- Caller — содержит ссылку на объект, который вызвал макрос;
- DisplayAlerts — управляет отображением диалоговых окон с предупреждениями;
- Evaluate — используется для расчета значения функций Excel и генерации объектов Range;
- InputBox — используется для запроса ввода у пользователя;
- OnKey — настраивает запуск макроса по нажатию клавиши или комбинации клавиш (в комбинации с <Ctrl>, <Alt> и т.д.);
- OnTime — используется для планирования запуска макроса в будущем;
- ScreenUpdating — управляет включением или отключением обновления экрана;
- Selection — содержит ссылку на выделенный диапазон;
- SendKeys — отправляет нажатия клавиш в активное окно;
- StatusBar — позволяет выводить сообщения в строке состояния;
- WorksheetFunction — содержит функции Excel, доступные для использования в VBA.

Это небольшой фрагмент общего количества свойств и методов объекта Application. В Excel 2003 предоставляется около 200 таких свойств и методов. Полный список приводится в приложении А.

4

Теория объектно-ориентированного программирования и VBA

Юлий Цезарь говорил: “Разделяй и властвуй”. Оригинальное утверждение относилось к ведению войны, в частности, к тактике разделения врага и победе над каждым батальоном в отдельности. Интересно, но этот результат относится и к победе над проблемами при проектировании программного обеспечения. Большие проблемы можно разделить на несколько маленьких и победить каждую из них. С другой стороны, можно позволить проблеме разделить наше внимание и победить нас.

Хорошим правилом является организация решения начиная с самой сложной и критической проблемы. Обычно эти фрагменты — основополагающие элементы решения, а приложение без этих фрагментов было бы неполным.

В этой главе рассматривается механизм, который обеспечивает разделение проблем и победу над ними — класс. В главе 5 будет рассматриваться общая теория объектно-ориентированного программирования, а также поддержка этой теории в языке VBA.

Сравнение классов и интерфейсов

Общая теория объектно-ориентированного программирования поддерживает два способа описания чего-то. Первым способом является класс. Обычно классы используются для описания сущностей в проблемной области. Крайне простым способом описания класса является его представление в виде существительного, имеющего значение в предметной области. Распространено представление класса в виде схемы, а объекта — в виде экземпляра сущности, которую описывает схема. Второй описательной конструкцией является интерфейс. Обычно интерфейс описывает возможности или определенный аспект сущности.

Для сравнения можно привести пример, в котором класс описывает телевизор, а интерфейс — возможности телевизора, например возможность увеличивать или уменьшать громкость. Но эта возможность может существовать не только в телевизоре. Например, увеличение громкости является аспектом всех сущностей, способных издавать звук. Автомобильные приемники, стереосистемы, телевизоры, проигрыватели, рожки и множество других предметов поддерживают возможность увеличения громкости. При этом между рожками и проигрывателями существует очень мало общего.

Как это связано с VBA? Ответ заключается в том, что модуль класса является гибридом между классом и интерфейсом. Если в модуле класса присутствуют только объявления, но отсутствует реализация (нет ни одной строки кода), то модуль является интерфейсом. Если в модуле класса присутствует реализация членов класса, то модуль класса является классом, реализующим интерфейс, точно соответствующий членам класса. (Важность реализации объявленных членов интерфейса заключается в том, что в некоторых объектно-ориентированных языках программирования класс может реализовывать несколько интерфейсов или ни одного, или некоторые члены, которые не определены в других интерфейсах.) По большей части модуль класса можно воспринимать как класс, но желательно помнить о существовании незаметного отличия.

И классы, и интерфейсы поддерживают полиморфизм. Это значит, что гибридный модуль класса также поддерживает полиморфизм. Полиморфизм классов поддерживается через наследование, возникающее при определении класса и второго подкласса, который генерализует и расширяет первый класс. Например, существует концепция класса `Элемент управления`, `Кнопка` является конкретным типом элемента управления. В терминах объектно-ориентированного программирования `Кнопка` является элементом управления. Наследование класса `Элемент управления` классом `Кнопка` называется генерализацией. Генерализация является синонимом наследования. Например, класс `Элемент управления` может поддерживать поведение `Щелчок`, а класс `Кнопка` расширяет это поведение, поддерживая изменение внешнего вида кнопки; пересмотренное поведение `Щелчок` является одной (из множества возможных, на что указывает слово полиморфизм) формой поведения `Щелчок`. Язык VBA не поддерживает наследование классов.

В языке VBA поддерживается полиморфизм интерфейсов. Полиморфизм интерфейсов ортогонален полиморфизму классов. При использовании классов можно объявить тип `Элемент управления` и инициализировать его с использованием класса `Кнопка`. При обращении к методу `Элемент управления.Щелчок` фактическое поведение `Щелчок` будет соответствовать поведению кнопки, так как `Кнопка` является `Элементом управления`. При полиморфизме интерфейсов объявляется тип интерфейса. Этому типу удовлетворяет любой класс, реализующий интерфейс. При этом никаких требований к связи между реализующими интерфейс типами не предъявляется. Рассмотрим классы `Собака` и `Шахматная фигура`. Эти классы не имеют ничего общего, но оба могут быть реализованы с возможностью `Перемещение`: `Собака.Перемещение` и `Ферзь.Перемещение`. Можно реализовать интерфейс `Перемещаемый` (`Imoveable` — префикс `I` используется в VBA при обозначении интерфейсов, а не модулей классов), в котором будет осуществлен метод `Перемещение`. Впоследствии любой код, использующий интерфейс `Перемещаемый` (`Imoveable`), сможет вызывать метод `Перемещение`, и фактический результат будет зависеть от конкретного объекта.

Технически полиморфизм классов поддерживает получение части поведения от подкласса. Например, вызов метода `Элемент управления.Щелчок` может приводить к возникновению события `OnClick`, а вызов метода `Кнопка.Щелчок` — к перекрашиванию кнопки и передаче управления родительскому коду, в котором создается событие. Полиморфизм интерфейсов скорее всего приведет к полной независимости от других

классов, реализующих интерфейс. Например, перемещение объекта `Собака` совершенно не повлияет на несвязанный объект `Шахматная фигура`.

Если планируется программировать на других языках, например Visual Basic .NET, стоит изучить незаметные различия между классами и интерфейсами. На данный момент оставим общие вопросы и перейдем к особенностям языка VBA.

Определение интерфейса

Язык VBA поддерживает определение интерфейса в одном модуле класса и реализацию интерфейса в другом модуле класса. Использование этого подхода позволяет реализовать один, два или больше классов, поддерживающих одинаковое поведение. Эта возможность полезна для доопределения методов в одном модуле класса.

Доопределение означает существование более чем одного метода с одним и тем же именем и разными сигнатурами аргументов. Доопределение методов в одном и том же классе невозможно, но можно воспользоваться полиморфизмом интерфейсов и определить несколько классов, предоставляющих одинаково именованное поведение.

Предположим, что необходимо определить класс, который рассчитывает факториал числа. Математически эта функция записывается как N!. Факториал является произведением всех целых чисел от 1 до N, где N является числом, для которого необходимо рассчитать факториал. Факториалы полезны при решении полиномиальных уравнений. (Реализацию функции факториала писать необязательно, так как можно воспользоваться встроенной в Excel функцией `Fact(n)`. Для демонстрации возможностей интерфейсов сделаем вид, что метод необходимо реализовать самостоятельно.) В редакторе VBE можно добавить модуль класса и объявить интерфейс `IFactorial`, в котором объявлен единственный метод `Calculate`. Вот соответствующий код метода:

```
Function Calculate(ByVal N As Long) As Long

End Function
```

В VBA модули классов с пустыми определениями являются чистыми интерфейсами. Если в окне Properties (Свойства) изменить значение свойства `Name` на `IFactorial`, то фактически будет объявлен чистый интерфейс `IFactorial`, в котором объявлен единственный метод `Calculate`. Метод `Calculate` принимает единственный параметр типа `Long` и возвращает значение типа `Long`. Теперь необходимо реализовать несколько версий этого интерфейса.

Реализация интерфейса

Для реализации интерфейса необходим второй модуль класса, в котором используется оператор `Implements`. Например, для реализации `IFactorial` из предыдущего раздела необходимо добавить второй и третий модули класса, добавив в каждый из них оператор `Implements IFactorial`. В каждом модуле класса необходимо предоставить реализацию метода `Calculate`. В следующих листингах приводятся примеры осуществления интерфейса `IFactorial`. В первом классе метод реализован в виде рекурсивной функции, а во втором классе — без рекурсии:

```
Implements IFactorial
Private Function IFactorial_Calculate(ByVal N As Long) As Long
  If (N > 1) Then
    IFactorial_Calculate = N * IFactorial_Calculate(N - 1)
  Else
```

```
    IFactorial_Calculate = 1
  End If
End Function

Implements IFactorial

Private Function IFactorial_Calculate(ByVal N As Long) As Long
  Dim I As Long, result As Long
  result = 1

  For I = 1 To N
    result = result * I
  Next I

  IFactorial_Calculate = result
End Function
```

В любом месте, где можно использовать интерфейс `IFactorial`, разработчику предоставляется возможность выбора конкретной реализации. В следующем листинге демонстрируется вызов второй реализации метода `Calculate`. Для использования второй реализации `IFactorial_Calculate` в операторе `Set` замените `Factorial1` на `Factorial2`:

```
Sub TestFactorial()

  Dim Factorial As IFactorial
  Set Factorial = New Factorial2
  MsgBox Factorial.Calculate(4)

End Sub
```

Естественно, что самостоятельное выполнение метода `Factorial` имеет смысл только при отсутствии удовлетворительной реализации. Как и в других ситуациях, придется решать, как определенную методику можно применить в конкретном случае.

В данном случае можно определить единственный модуль класса, в котором определены методы `RecursiveFactorial` и `LoopFactorial` (или подобные методы), но объявление интерфейса `IFactorial` и пары вариантов метода упрощает выбор поведения за счет полиморфизма. Использование интерфейса позволит менять только оператор создания объекта, реализующего интерфейс, и не модифицировать все вызовы методов, так как вызовы сохранят форму `объект.метод`, как показано в предыдущем примере. Если используется один класс и два метода, то замена метода потребует модификации всех вызовов в коде. Другими словами, при применении интерфейсов код меняется только в одном месте — в объявлении, а при использовании старого способа с применением нескольких имен код придется модифицировать в нескольких местах.

Кто-то может спорить, утверждая, что в данном случае гибкость использования интерфейсов практически незаметна, но успешное программирование является скоординированным применением лучших из лучших практик. В итоге, используя лучшие практики, в данном случае интерфейсы, можно получить более качественный общий результат.

Определение методов

Методы описывают поведение класса. Все действия класса должны быть реализованы в виде подпрограмм или функций в пределах класса. Подпрограммы и функции в классах называются методами. Хорошие имена методов обычно состоят из целого глагола и существительных. Если необходимо возвращать единственное значение, реализуйте функцию. Если необходимо выполнить определенную задачу без возврата значения, реализуйте подпрограмму.

Объявление функции состоит из модификатора доступа, Friend, Public, Private (дополнительная информация приводится в разделе "Сокрытие информации и квалификаторы доступа" далее в этой главе), необязательного ключевого слова Static, ключевого слова Function, имени функции, необязательных параметров и типа возвращаемого значения. Объявление функции состоит из модификатора доступа, необязательного ключевого слова Static, ключевого слова Sub, имени и необязательных параметров.

Выбор между функцией или подпрограммой, выбор количества и типа параметров, а также типа возвращаемого значения (если это функция) определяется практическими соображениями. Вот несколько хороших правил:

- используйте комбинации глаголов и существительных, а также целые слова для составления имен методов;
- передавайте только совершенно необходимые параметры. Если параметр является полем того же класса, что и метод, его можно не передавать в качестве параметра;
- используйте целые слова, желательно существительные, для обозначения параметров;
- осторожно выбирайте модификаторы доступа, стараясь максимально сократить количество методов с модификатором Public (дополнительная информация приводится далее в этой главе в разделе "Сокрытие информации и квалификаторы доступа").

Синтаксис объявления методов приводится в справочном руководстве. Кроме этого, в этой и в других главах рассматривается множество примеров объявлений.

Аргументы

Аргументы могут передаваться по значению, по ссылке и в виде необязательного параметра. Аргументы по значению определяются с помощью ключевого слова ByVal. Присутствие ByVal означает, что изменение значения аргумента незаметно за пределами метода, так как в метод передается копия оригинального значения. Аргументы по ссылке определяются с помощью ключевого слова ByRef. Аргументы после ByRef являются ссылками на значения параметров; модификация передаваемых по ссылке параметров отражается за пределами метода. Наконец, модификатор Optional может использоваться вместе с ключевыми словами ByVal и ByRef. Необязательные аргументы должны следовать после обязательных.

Передача аргументов по значению

Аргументы ByVal могут меняться в пределах метода, но это изменение не отражается за его пределами. Например, если определить такую процедуру:

```
Sub DoSomething(ByVal I As Integer)
  I = 10
End Sub
```

и вызвать процедуру DoSomething из следующего кода:

```
Dim J As Integer
J = -7
DoSomething(J)
' J все еще равно -7
```

После завершения работы процедуры DoSomething значение аргумента J все еще будет равно -7. Необходимо понять механизмы управления состоянием приложения, что позволит успешно это состояние контролировать. Другими словами, передаваемый по значению (ByVal) параметр меняется только в пределах метода.

Передача аргументов по ссылке

Если не указывать квалификатор аргумента, по умолчанию аргументы передаются по ссылке. Квалификатор ByRef по сути определяет указатели на переменные. Таким образом, при модификации процедуры DoSomething для использования ссылок на аргументы после завершения работы метода переменная J будет иметь значение 10.

Чтобы понять, как обрабатываются аргументы, передаваемые по ссылке и по значению, стоит помнить, что любой объект является числом. Переменные представлены двумя числами. Первое определяет положение переменной в памяти. Второе — значение переменной. При передаче аргумента по значению передается два числа. Первое указывает расположение копии переменной, а второе — значение копии переменной. При передаче аргумента по ссылке передается расположение переменной, а значит используется один экземпляр значения.

Необязательные аргументы

Вот основной принцип использования необязательных параметров: если определить параметр, который большую часть времени имеет определенное значение, разработчик экономит время потребителей. Например, если определить класс, рассчитывающий налог с продаж, а приложение в основном используется в штате Мичиган, то можно определить необязательный параметр Tax, который по умолчанию имеет значение в 6 процентов:

```
Public Function AddSalesTax(ByVal SaleAmount As Double, _
  Optional ByVal Tax AS Double = .06) As Double
  AddSalesTax = SaleAmount * (1 + Tax)
End Sub
```

> Под потребителем подразумевается пользователь создаваемого кода. Например, если класс создается и используется одним и тем же человеком, то программист является потребителем класса. Таким образом, программист экономит собственное время.

В соответствии с определением процедуру AddSalesTax можно вызывать с одним параметром SaleAmount или с параметрами SaleAmount и Tax. Например, если метод используется в штате Орегон, в качестве значения параметра Tax можно передать 0, так как в штате Орегон отсутствует налог с продаж.

Реализация рекурсивных методов

Рекурсивные методы вызывают сами себя в процессе работы. Обычно в пределах метода присутствует условие завершения работы. Рекурсивные функции, например Factorial, рассматривались ранее в этой главе. Функции такого типа можно реализовать быстро и просто. Единственная проблема заключается в развертывании стека.

При вызове метода адрес текущей процедуры записывается в стек, который является ограниченным ресурсом. После этого в стек добавляются параметры, и метод вызывается еще раз. Адрес текущей процедуры используется для возврата после завершения работы метода. После вызова в стек добавляются локальные переменные. Таким образом, при каждом вызове метода в стек добавляется информация, которая не извлекается до завершения работы метода.

Рекурсивные методы вызывают сами себя до завершения своей работы, складируя информацию в стеке. Если количество итераций оказывается слишком большим, память стека может закончиться, взорвав стек. Если такая ситуация возникает в VBA, приложение выдаст сообщение об ошибке времени выполнения 28, "Out of stack space". Из-за вероятности взрыва стека рекурсивные методы недостаточно жизнеспособны. Обычно для избавления от рекурсии используются циклы.

Отказ от рекурсии через использование циклов

Рекурсивные методы вызывают сами себя в процессе решения задачи. Вызов самого себя является неявным циклом, а условие завершения – граничным условием цикла. Например, рекурсивный метод Factorial вызывает себя до момента, когда N будет равно 1. После этого работа метода завершается. Это значит, что граничными условиями являются 1 и N. Так как 1 * M равно M, 1 тоже можно исключить из условия. Это значит, что граничными условиями являются 2 и N. Так как для использования цикла необходимо знать верхнюю и нижнюю границы, можно реализовать рекурсивный алгоритм подсчета факториала с помощью цикла.

Существует несколько причин, по которым рекурсию лучше заменять на циклы: первая причина заключается в большей скорости цикла FOR по сравнению с разворачиванием стека и вызовом метода; вторая причина в том, что цикл может выполняться бесконечное количество раз, а объем стека рано или поздно закончится. Следовательно, избавляйтесь от рекурсивных методов, так как они могут стать причиной проблем у конечных пользователей.

Определение полей

Поля описывают состояние объекта. Они могут принадлежать любому типу. Желательно предоставлять принятое по умолчанию начальное значение поля в методе Class_Initialize, а принятое по умолчанию конечное значение в методе Class_Terminate. Инициализация полей позволяет обеспечить известное состояние объекта в начале жизни. Добавление финального состояния является хорошей привычкой, которая помогает избежать использования уничтоженных объектов.

Например, если объект имеет строковое поле MyField и в процедуре Class_Terminate полю присваивается значение "done", перед использованием объекта всегда можно проверить значение поля MyField и сравнить его со строкой "done". Хотя этот прием оказывается более полезным в таких языках, как C++, C и Object Pascal, управление состоянием является наиболее важной привычкой.

Сигнатурой поля является ключевое слово Private (по соглашению все поля являются закрытыми), имя поля и ключевое слово As, после которого указывается тип поля. Поля являются закрытыми, так как неограниченный доступ к состоянию усложняет управление состоянием объекта. Проблема полей заключается в том, что большинство из них могут иметь конечное множество значений. Например, существует конечное число действительных почтовых индексов. Это значит, что необходимо обеспечить присвоение полям конечного количества действительных значений, чтобы обеспечить действительные начальное и последующие состояния. Сложность заключается в том, что само по себе поле не может вызывать код. Для этого существуют свойства.

Определение свойств

Хотя поля — это неотъемлемые компоненты управления состоянием, любое поле, предоставляемое потребителю, должно сопровождаться свойством. Свойство является специальным методом, который используется как поле, но на самом деле выполняет определенный код. В результате свойства можно использовать вместо полей, получая удобный механизм управления конечным множеством действительных значений поля.

Свойства имеют специальную сигнатуру, но выглядят как методы. Сигнатура свойства состоит из модификатора доступа (обычно `Public`), ключевого слова `Property`, ключевого слова `Get`, `Let` или `Set`, имени свойства и тела функции или подпрограммы. Для определения свойства, возвращающего значение, используется ключевое слово `Get`, скобки, ключевое слов `As` и тип возвращаемого значения. Для определения свойства, которое устанавливает значение, применяются ключевое слово `Set` или `Let` и параметр для типа устанавливаемого значения. Если значение записывается в поле классового типа, используется ключевое слово `Set`. Если поле имеет неклассовый тип, применяется ключевое слово `Let`. Например, если существует поле для хранения имени, можно определить следующую комбинацию полей и свойств:

```
Private FFirstName As String

Public Property Get FirstName() As String
    FirstName = FFirstName
End Property

Public Property Let FirstName(ByVal Value As String)
  FFirstName = Value
End Property
```

По соглашению префикс `F` используется для обозначения полей. Для получения имени свойства достаточно убрать префикс `F`. Полученное имя позволит легко определить соответствующее свойству поле. (Некоторые программисты применяют префикс в виде аббревиатуры названия типа, но изобретатель этого стиля, компания Microsoft, рекомендует от этого отказаться. В данном случае каждый решает сам для себя. Имя `FirstName` с использованием префикса может быть записано как `sFirstName`, `m_FirstName` или `mvar_FirstName`.) Свойство `Get` возвращает значение поля. Свойство `Let` присваивает значение аргумента `Value` (по соглашению это имя используется для обозначения аргументов свойства) полю объекта. Если необходимо написать код проверки действительности значения, он должен быть добавлен между первой и последней строками оператора свойства.

Обычно код проверки действительности присутствует только в методах свойств `Let` и `Set`. В первом листинге показано, как проверить действительность значения свойства налога с продаж, который должен быть неотрицательным и не должен быть меньше 10%. Во втором листинге показано, как можно создавать составное свойство `FullName` на основе полей имени и фамилии:

```
Private FSalesTax As Double

Public Property Let SalesTax(ByVal Value As Double)
  If (Value < 0 Or Value > 0.1) Then
    ' Сделать что-то, чтобы сообщить об ошибке
  End If

  FSalesTax = Value
End Property
```

В предыдущем примере продемонстрировано, как проверять действительность значения свойства `SalesTax`. Можно предположить, что налог не будет превышать 10% еще в течение нескольких месяцев, а отрицательных налогов не существует.

```
Public Property Get FullName() As String
  FullName = FFirstName & " " & FLastName
End Property
```

В данном случае показано, как можно создавать составное свойство. Свойство `FullName` соответствует полям `FFirstName` и `FLastName` (не забывайте о соглашении по использованию префикса `F`). При желании можно написать свойство `Let FullName`, которое будет делить строку на две подстроки и присваивать результат полям `FFirstName` и `FLastName`.

Помните, что свойства на самом деле являются специальными методами. Хотя в теле свойства можно написать любой код, основным назначением свойства является контроль допустимости данных. Это значит, что свойство отвечает за надежность установки и возврата значений полей.

Свойства только для чтения

Предназначенное только для чтения свойство имеет лишь определения `Property Get` (без определения методов `Set` и `Let`). Если используется составное свойство или неизменяемое поле, которое потребители могут читать, но не должны менять его значение, определите только метод `Property Get`.

Предназначенные исключительно для чтения свойства имеют только метод `Get`, но синтаксис этого метода не отличается от методов обычных свойств.

Свойства только для записи

Свойства только для записи имеют методы `Property Set` и `Property Let`, но не имеют метода `Property Get`. (`Let` используется для записи в необъектные переменные, а `Set` — для записи в объектные переменные.) Свойства только для записи встречаются достаточно редко, но время от времени используются. Например, бензин можно заливать в бак автомобиля через крышку, но если нет желания набирать бензин в рот, бензин невозможно удалить через то же отверстие, через которое он был залит. Только езда на автомобиле приведет к снижению уровня бензина в баке. Другими словами, свойства только для чтения технически возможны, но практически используются очень редко.

Определение событий

Операционная система Windows и Excel, как одно из приложений, построены на базе троицы состояния, поведения и сигналов. Составляющие троицы реализуются идиомами состояния и свойства, метода и события. События являются способом, которым объекты сообщают получателям, что выполнено определенное действие и сейчас самое подходящее время прореагировать.

Не стоит предполагать, что отличное внутреннее проектирование Excel обеспечивает возможность программирования приложений Excel без понимания механизмов событий. На самом деле, если досконально изучить механизмы событий, станет очевидным их применение в приложениях. Поэтому здесь приводится концептуальное описание событий: производители (которые создают классы) не всегда наперед знают, как потребители (которые используют классы) будут реагировать на изменение состояния, поэтому в такой ситуации используются события, предоставляющие получателям неограниченное

пространство для реакции на изменение состояния. За кулисами синтаксиса событие является не более чем неинициализированным числом, которое указывает на метод. Когда программист связывает обработчик события (обычный метод) с событием, оно больше не является неинициализированным числом. Теперь событие имеет номер (который называется адресом) обработчика событий.

При определении метода в класс добавляется предварительно инициализированное число, которое ссылается на определяемый метод. При определении события добавляется неинициализированное число, которое позволяет потребителю добавить код в точке возникновения события. Таким образом, события являются зарезервированными местами для добавления необходимых потребителям ответов.

Перед производителем стоит две задачи: определение событий, выступающих в роли сигнатур точек подключения, а также генерация событий в точках, где потребителям разрешено подключать собственный код. С точки зрения потребителя, событие — это возможность реагировать на изменение состояния объекта. Например, при изменении значения свойства `TextBox` потребитель может проверить, является ли значение числом от 1 до 9 или строкой в формате почтового кода США. Производитель не может заранее знать о необходимой потребителю функциональности. Следовательно, он программирует универсальную функциональность, а потребитель конкретизирует ее для решения собственных задач.

Определение событий в классах

Не существует двух одинаковых типов реализаций объектных языков. Изучение нескольких языков позволяет получить общее представление о доступных возможностях. Например, идиома событий в VBA ограничена модулями классов, формами и документами, но так как событие является указателем на функцию, напрашивается вывод, что это ограничение налагается самим языком VBA.

Кроме ограничения определения и использования событий в пределах модулей классов, форм и документов, события в VBA должны объявляться как подпрограммы и не могут применять именованные и необязательные параметры, а также параметры `ParamArray`. События можно определять и использовать внутри классов (формы и документы являются специальными классами). Определение событий в обычном модуле не поддерживается. (В других языках идиома указателей на функции не связана такими ограничениями. С этой точки зрения больше всего возможностей доступно в C++.)

Определять событие в классе имеет смысл там, где потребитель может вставить реакцию на изменение состояния класса. В определении события используется модификатор доступа (обычно `Public`), ключевое слово `Event` и заголовок подпрограммы, где ключевое слово `Sub` неявно подразумевается под ключевым словом `Event`. В следующем примере показано, как определить событие для потребительского калькулятора. Это событие предоставляет точку вставки, где потребители класса `Calculator` могут реагировать на изменение потребительского параметра `Discount` (одним из вариантов реакции является пересчет общей суммы):

```
Public Event DiscountChanged (ByVal sender As Calculator, _
       ByVal NewDiscount As Double)
```

Как показано в объявлении, используются ключевые слова `Public` и `Event`. После них указывается определение подпрограммы с аргументами, но без типа возвращаемого значения. В этом примере есть ссылка на объект, содержащий событие. (Полезность этого приема доказана во множестве объектно-ориентированных языков, поэтому здесь эта практика также применяется.) В качестве второго аргумента передается значение `NewDiscount`.

Создание события

Процесс создания события по соглашению состоит из двух этапов: вызова метода с квалификатором доступа `Private` и создания события этим методом. Использование метода-оболочки обеспечивает конкретное место для реализации дополнительного поведения, а для самого создания события применяется оператор `RaiseEvent`. В следующем примере показано, как создавать событие `DiscountChanged`.

```
Private FDiscount As Double

Public Property Get Discount() As Double
  Discount = FDiscount
End Property

Public Property Let Discount(ByVal value As Double)
  If (value <> FDiscount) Then
     FDiscount = value
     Call DoDiscountChanged(FDiscount)
  End If
End Property

Private Sub DoDiscountChanged(ByVal NewDiscount As Double)
   RaiseEvent DiscountChanged(Me, NewDiscount)
End Sub
```

В этом листинге представлены все элементы, имеющие значение в процессе создания и использования события. Первый оператор определяет поле, в котором хранится величина скидки. Оператор `Property Get` позволяет получить текущее значение скидки, а оператор `Property Let` проверяет, стоит ли выполнять код изменения поля (если значение скидки не меняется, то соответствующий код не выполняется). Условный оператор в свойстве `Property Let` является просто соглашением. Если значение скидки все-таки меняется, свойство вызывает метод `DoDiscountChanged`, который объявлен с квалификатором доступа `Private`. Наконец, метод `DoDiscountChanged` создает событие. На самом деле для создания события необходим только оператор `RaiseEvent`, но использование метода-оболочки позволяет отказаться от передачи первого аргумента — ссылки на себя.

Соглашения принимаются одним из двух способов: тяжелым методом проб и ошибок или через копирование опыта тех, кто уже прошел по этому пути. Соглашение по сокрытию создания событий позволяет локализовать любой зависящий от внешних факторов код. Например, предположим, что принято решение о сохранении величины скидки в базе данных, файле или системном реестре. Вместо добавления этого кода во всех точках, где должно создаваться событие, код добавляется только в метод `DoDiscountChanged`.

Наконец, рассмотрим вопросы обеспечения производительности и качества. Следование соглашениям, например сокрытию создания события, даже если это не всегда имеет смысл, позволяет получать однородный и симметричный код, который сам по себе является признаком профессионализма. При этом потребители и другие разработчики кода сразу поймут, чего от него ждать. Кроме этого, делая одно и то же в одинаковых обстоятельствах, разработчик экономит время, не думая о том, как и почему что-то делается именно так. В конечном результате надежные привычки формируются намного быстрее и разработчик может уделять основное внимание проблеме и ее решению, а не обдумыванию методологии написания кода.

Обработка событий

При определении события разработчик выполняет роль производителя. При создании кода обработчика события разработчик выполняет роль потребителя. Роль производителя и потребителя в одном лице несет в себе определенный риск, так как потребитель, одновременно являясь производителем, обладает знанием о внутреннем поведении потребляемого класса и может попытаться "срезать напрямик", например, записывая код непосредственно в поле, вместо создания и использования события. Оптимальным решением является создание классов для среднего потребителя в соответствии с правилами хорошего производителя, после чего, даже будучи потребителем собственного кода, можно создать хороший код, предполагая, что о внутреннем устройстве потребляемых классов ничего неизвестно.

Для обработки события с помощью ключевого слова `WithEvents` необходимо объявить переменную и использовать редактор кода для генерации обработчика события. В следующем листинге показано, как в пределах листа можно определить экземпляр класса `Calculator` и выбрать класс и событие из раскрывающихся списков **Object** (Объект) и **Procedure** (Процедура):

```
Private WithEvents theCalculator As Calculator

Public Sub TestCalculator()

  Set theCalculator = New Calculator
  theCalculator.Discount = 0.05

End Sub
Private Sub theCalculator_DiscountChanged(ByVal sender As Calculator, _
  ByVal NewDiscount As Double)

  Call MsgBox("Скидка изменилась на " & NewDiscount)

End Sub
```

Первый оператор определяет переменную калькулятора с помощью ключевого слова `WithEvents`. При этом переменная `theCalculator` будет добавлена в список объектов в редакторе кода. Выберите `theCalculator` из списка объектов, тогда в списке процедур станет доступна процедура `DiscountChanged`. Так как это единственное событие, в модуль потребителя будет добавлена процедура `theCalculator_DiscoundChanged`.

В этом примере просто выводится новая величина скидки. В реальном приложении может потребоваться пересчет итоговой суммы или подобная операция. Обычно лучше добавлять события во все места, где потребителю потребуется среагировать на изменение состояния. Помните, что разработчик как производитель, не в состоянии предусмотреть все существующие и будущие применения класса. Использование событий позволяет увеличить вероятность успешного применения класса в новых условиях, что увеличивает срок жизни кода без дополнительной модификации.

Сокрытие информации и квалификаторы доступа

На протяжении этой главы несколько раз упоминались квалификаторы доступа. В языке программирования VBA квалификаторами доступа являются ключевые слова `Private`, `Public` и `Friend`. Перед тем как рассматривать значение конкретных ключевых слов, отметим, что в VBA все сущности можно определить с квалификатором `Public`, что

особенно полезно в самом начале знакомства с языком. Но используя только открытый доступ, разработчик отказывается от одной из самых полезных, хотя и сложных, возможностей объектно-ориентированного программирования.

Перед знакомством с различными аспектами квалификаторов доступа разберемся, что эти слова означают на практике. И потребитель, и производитель могут вызывать открытые (public) методы. Открытые члены класса похожи на общественные туалеты. Любой может войти и делать что угодно. Закрытые (private) члены класса доступны только производителю. Только производитель класса может обращаться к закрытым членам. Друзья (Friend) могут только модифицировать процедуры в модулях `Form` и `Class` и не поддерживают позднюю привязку. Квалификатор `Friend` означает, что процедура является открытой в пределах проекта и закрытой для внешних проектов. С точки зрения Excel это значит, что процедуры `Friend` можно вызывать из любого места в пределах книги, но не из внешних книг.

Зачем нужны квалификаторы и в чем их польза? В основном, чем меньше в классе открытых членов, тем он проще в использовании. Это значит, что открытыми должны быть только несколько необходимых членов, а все остальные должны быть объявлены, как закрытые. Это очень субъективное мнение. С другой стороны можно рассмотреть следующее обстоятельство: если член является закрытым, превращение его в открытый не сделает зависящий от него код неработоспособным, но закрытие открытого члена может привести к неработоспособности зависящего от него кода. Вот основные рекомендации, которым необходимо следовать, принимая решение об использовании квалификаторов доступа `Public` и `Private`:

- старайтесь предоставлять не более пяти открытых членов;
- если сложно принять решение, сделайте член закрытым и проверьте его способность выполнять поставленную задачу;
- все поля должны быть закрытыми;
- все свойства должны быть открытыми;
- все события должны быть открытыми;
- если возникает необходимость в более чем 6 открытых членах, рассмотрите возможность разбиения класса;
- при необходимости нарушайте любое правило; никто за это не наказывает.

Инкапсуляция, агрегация и ссылки

Завершим главу кратким обсуждением полезных, но сложных для понимания концепций. Имеются в виду инкапсуляция, агрегация, ссылки и зависимости. Инкапсуляция описывает хранение чего-то, агрегация — составление целого из фрагментов; ссылки — знание чего-то одного и использование чего-то другого, а зависимости говорят сами за себя.

В объектно-ориентированном мире говорят "инкапсуляция", а подразумевают "хранящий данные класс". Например, класс `Workbook` инкапсулирует класс `Worksheets`. Инкапсуляция обычно проявляется именем переменной, членом оператора и инкапсулированной информацией. Например, лист инкапсулирует ячейки. Их связь в классе `Worksheet` может быть представлена в виде свойства `Cells`. Тогда код будет выглядеть, как `Sheet1.Cells`.

Агрегация описывает целое, части и сумму этих частей. Автомобиль состоит из колес и шин, двигателя и трансмиссии, сидений и руля. При соединении частей формируется новый предмет. Смысл агрегации заключается в том, что, при наличии частей и использовании различных способов их комбинирования, в итоге можно получать разный результат (целое). Автомобиль Hummer не является автомобилем Jeep, но они могут использовать одинаковые детали. Как минимум, они используют одинаковые концепции.

Самолет не является автомобилем, но и самолеты, и автомобили имеют общие компоненты, например двигатели внутреннего сгорания и шины. Агрегация напоминает, что комбинирование частей позволяет получить более мощный результат, чем при использовании единичного монолитного предмета.

Концепция ссылки описывает знание одного класса о другом классе. Например, экземпляр класса `Calculator` можно добавить в класс `Worksheet`. В таком случае лист будет отвечать за создание калькулятора. Кроме этого, за создание калькулятора может отвечать другой класс, а класс `Calculator` будет просто ссылаться на созданный калькулятор. Если объект класса `Worksheet` создает объект класса `Calculator`, то объект класса `Calculator` является частью целого и такое отношение называется агрегацией; если объект класса `Worksheet` просто ссылается на объект класса `Calculator`, то такое отношение называется ссылкой. Хорошим примером отношения ссылки является объект `Application`. Многие классы в Excel имеют свойство `Application`, но при удалении объекта `Worksheet` объект `Application` не исчезает. Но если приложение Excel закрывается, исчезает и объект `Worksheet` (хотя содержимое объекта и сохраняется в файле). Таким образом, отношение между объектами `Application` и `Worksheet` является агрегацией, а между `Worksheet` и `Application` — ссылкой.

Наконец, существует отношение зависимости, при котором существование одного объекта зависит от существования другого. Объект `Worksheet` зависит от существования объектов ячеек. Если класс A зависит от другого класса B, то объект класса A не может функционировать без объектов класса B. Такое отношение зависимости существует между объектом `Application` (приложение Excel) и объектом `Worksheet`. Приложение Excel реализовано таким образом, что на экране должен быть виден как минимум один лист. Следовательно, всегда можно положиться на истинность условия `ThisWorkbook.Sheets.Count >= 1`. Определение и использование зависимостей основано на надежности условий: где существует один объект, там должен существовать и второй.

Эти термины приводятся здесь, так как они будут использоваться на протяжении всей книги. Так как это универсальные объектно-ориентированные концепции, они полностью справедливы и для VBA. Понимание этой терминологии позволит вести более плодотворное обсуждение и сделает понятными многие расширенные методы программирования, которые описываются в специализированных и общих книгах по объектно-ориентированному программированию.

Резюме

Excel является практическим инструментом, неплохо справляющимся с математическими задачами и графическим представлением решений этих задач. Но VBA также является мощным языком программирования и подходит для решения задач общего характера. VBA почти ничем не отличается от Visual Basic, а язык Visual Basic позволяет решать практически любые задачи. Рассмотрев универсальные объектно-ориентированные концепции, можно наиболее полно использовать возможности Excel и обратиться к более сложным концепциям. Если Excel необходима для решения проблем, требующих применения других приложений Office или Visual Basic, эти знания могут оказаться исключительно полезными.

Если возникла необходимость более глубоко изучить язык программирования VBA, рекомендуем познакомиться с книгами о *шаблонах (patterns)* и *рефакторинге (refactoring)*, например *Add Refactoring: Improving the Design of Existing Code* Мартина Фаулера (Martin Fowler) или *Design Patterns* Эрика Гамма (Erich Gamma) и др. С объектно-ориентированной точки зрения Excel отлично справляется с "перемалыванием" чисел, а язык VBA является не просто макроязыком.

5

Процедуры обработки событий

Excel значительно облегчает создание кода, который обрабатывает событие листа, листа диаграммы или книги. В предыдущих главах было показано, как подсветить активную строку или столбец листа, добавив код в процедуру обработки события Worksheet_SelectionChange (глава 2). Эта процедура запускается каждый раз, когда пользователь выделяет новый диапазон ячеек. (Кроме этого, в главе 19 будут представлены примеры синхронизации листов в книге с помощью событий Worksheet_Deactivate и Worksheet_Activate.)

События книг, листов диаграмм и листов создавать очень легко, так как Excel автоматически предоставляет модули кода для этих объектов. Но стоит обратить внимание, что предоставляемые автоматически события диаграмм в модуле диаграммы относятся только к листам диаграмм и никак не связаны со встроенными диаграммами. Для создания процедур обработки событий встроенных диаграмм потребуются дополнительные знания и трудозатраты.

Кроме этого, существует множество других высокоуровневых событий, к которым может получить доступ разработчик, например события объекта Application. Они рассматриваются далее в главах 6 и 14. События элементов управления и форм также анализируются в соответствующих главах. В этой же главе более подробно будут рассмотрены события листов, диаграмм и книг, а также — проблемы, связанные с использованием этих событий.

> Процедуры обработки событий всегда связываются с определенным объектом и хранятся в модуле класса, который соответствует данному объекту. Например, это может быть модуль объекта ThisWorkbook или модуль кода, связанный с листом или диалоговым окном UserForm. События определяются только в модулях классов.

События листа

Следующие процедуры обработки событий листа доступны в модуле кода, связанном с каждым листом:

- Private Sub Worksheet_Activate()
- Private Sub Worksheet_BeforeDoubleClick(ByVal Target As Range, Cancel As Boolean)
- Private Sub Worksheet_BeforeRightClick (ByVal Target As Range, Cancel As Boolean)
- Private Sub Worksheet_Calculate()
- Private Sub Worksheet_Change(ByVal Target As Range)
- Private Sub Worksheet_Deactivate()
- Private Sub Worksheet_FollowHyperlink (ByVal Target As Hyperlink)
- Private Sub Worksheet_PivotTableUpdate(ByVal Target As PivotTable)
- Private Sub Worksheet_SelectionChange(ByVal Target As Range)

Для создания шаблона процедуры обработки события можно воспользоваться раскрывающимся списком в верхней части модуля кода. Например, в модуле кода листа можно выбрать объект Worksheet из левого раскрывающегося списка. Это приведет к генерации следующего кода:

```
Private Sub Worksheet_SelectionChange(ByVal Target As Range)
...
End Sub
```

Событие SelectionChange является принятым по умолчанию для объекта Worksheet. Из правого раскрывающегося списка можно выбрать другое событие. После этого необходимо удалить из модуля строки кода из предыдущего примера.

Вместо использования раскрывающихся списков строки определения процедуры можно ввести вручную. Тип процедуры, а также тип, количество и порядок аргументов (так называемая сигнатура) должны совпадать с предыдущим кодом. При необходимости используйте другие имена параметров, но лучше – стандартные имена, что позволит избежать ненужной путаницы.

Большинство параметров процедур обработки событий должны объявляться с квалификатором ByVal, что защищает соответствующие объекты или сущности от модификации в процессе работы процедуры. Если параметр является объектом, можно использовать методы этого объекта и менять его свойства, но обратная передача определения объекта через присваивание нового определения параметру невозможна.

Некоторые процедуры обработки событий выполняются до возникновения события и имеют параметр Cancel, который передается по ссылке (ByRef). Параметру Cancel можно присвоить значение True, что приводит к отмене данного события. Например, пользователю можно запретить доступ к контекстному меню листа. Для этого в процедуре Worksheet_BeforeRightClick нужно отменить событие RightClick:

```
Private Sub Worksheet_BeforeRightClick (ByVal Target As Range, _
        Cancel As Boolean)
   Cancel = True
End Sub
```

Включение событий

В некоторых процедурах обработки события необходимо отключать. Это позволяет избежать неявной рекурсии. Например, если процедура обработки события листа Change меняет его содержимое, она сама будет приводить к возникновению события Change, что вызовет повторный запуск процедуры. Процедура обработки событий еще раз изменит содержимое листа и еще раз создаст событие Change и т.д.

Если в такой рекурсии принимает участие только одна процедура, Excel 2000, 2002 и 2003 обнаружит рекурсию и завершит выполнение процедуры после нескольких сотен повторений (Excel 2003 повторит выполнение процедуры Change после 226 повторений, а Excel 97 прекратит после 40 повторений.) Если в рекурсии принимает участие более одной процедуры обработки событий, процесс будет продолжаться неограниченное время, пока пользователь не нажмет клавишу <Esc> или комбинацию клавиш <Ctrl+Break> столько раз, сколько было запущено процедур обработки событий.

Например, могут быть активны процедуры обработки событий Calculation и Change. Если обе процедуры меняют ячейку, на которую ссылаются расчеты, обе процедуры обработки события будут вызываться в процессе интерактивной цепной реакции. То есть, первая процедура обработки события приводит к вызову второй процедуры, а вторая — к вызову первой и т.д. Следующая процедура обработки события Change защищается от цепной реакции, отключая обработку событий во время изменения содержимого листа. Ее необходимо включить после завершения работы процедуры:

```
Private Sub Worksheet_Change(ByVal Target As Range)
    Application.EnableEvents = False
    Range("A1").Value = 100
    Application.EnableEvents = True
End Sub
```

> Оператор Application.EnableEvents = False не влияет на события за пределами объектной модели Excel. Например, события элементов управления ActiveX и диалоговых окон UserForm продолжают обрабатываться.

Событие Worksheet Calculate

Событие Worksheet_Calculate возникает при пересчете содержимого листа. Обычно это событие создается при вводе новых данных в ячейки, на которые ссылаются формулы листа. Например, при вводе прогнозируемых значений данных процедуру обработки события Worksheet_Calculate можно использовать для выдачи предупреждения, когда ключевые результаты выходят за пределы допустимого диапазона. На листе, показанном на рис. 5.1, предупреждение выдается каждый раз, когда значение прибыли в ячейке N9 оказывается больше 600 или меньше 500.

Следующая процедура обработки события запускается каждый раз при пересчете содержимого листа, проверяет содержимое ячейки N9 (которая называется FinalProfit) и генерирует сообщения в случае выхода значения за пределы указанного диапазона.

```
Private Sub Worksheet_Calculate()
   Dim Profit As Double

   Profit = Sheet2.Range("FinalProfit").Value
   If Profit > 600 Then
      MsgBox "Прибыль увеличилась до " & Format(Profit, "#,##0.0")
   ElseIf Profit < 500 Then
```

```
        MsgBox "Прибыль снизилась до " & Format(Profit, "#,##0.0")
    End If
End Sub
```

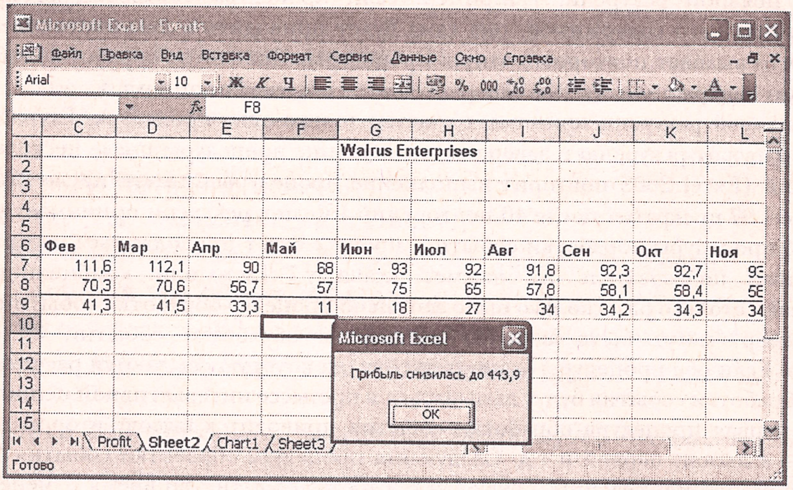


Рис. 5.1. Контроль меняющегося значения

События диаграммы

Следующие процедуры обработки событий доступны в модуле кода каждого объекта диаграммы.

- `Private Sub Chart_Activate()`
- `Private Sub Chart_BeforeDoubleClick(ByVal ElementID As Long, ByVal Arg1 As Long, ByVal Arg2 As Long, Cancel As Boolean)`
- `Private Sub Chart_BeforeRightClick(Cancel As Boolean)`
- `Private Sub Chart_Calculate()`
- `Private Sub Chart_Deactivate()`
- `Private Sub Chart_DragOver()`
- `Private Sub Chart_DragProt()`
- `Private Sub Chart_MouseDown(ByVal Button As XlMouseButton, ByVal Shift As Long, ByVal x As Long, ByVal y As Long)`
- `Private Sub Chart_MouseMove(ByVal Button As XlMouseButton, ByVal Shift As Long, ByVal x As Long, ByVal y As Long)`
- `Private Sub Chart_MouseUp(ByVal Button As XlMouseButton, ByVal Shift As Long, ByVal x As Long, ByVal y As Long)`
- `Private Sub Chart_Resize()`
- `Private Sub Chart_Select (ByVal ElementID As XlChartItem, ByVal Arg1 As Long, ByVal Arg2 As Long)`
- `Private Sub Chart_SeriesChange(ByVal SeriesIndex As Long, ByVal PointIndex As Long)`

Событие BeforeDoubleClick

Обычно при двойном щелчке на элементе диаграммы открывается диалоговое окно с параметрами форматирования элемента. Перехват события двойного щелчка с созданием собственного кода позволяет сократить процедуру настройки форматирования.

В следующей процедуре форматирование трех элементов диаграммы выполняется при двойном щелчке на элементе. Если выполнить двойной щелчок на легенде диаграммы (рис. 5.2), легенда исчезает.

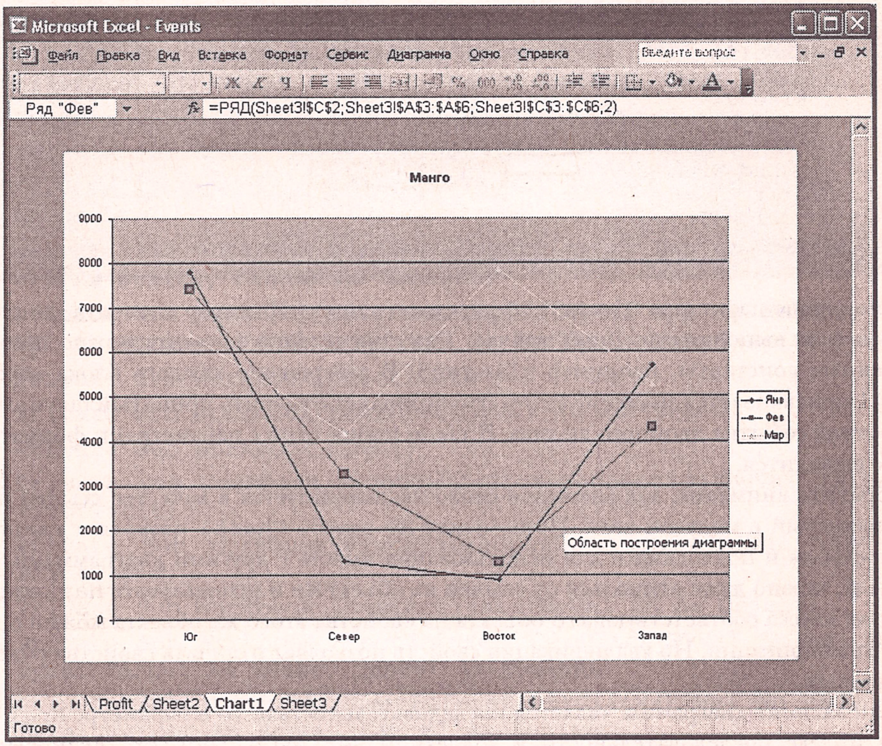


Рис. 5.2. Обработка событий диаграммы

Если выполнить двойной щелчок на области диаграммы (за пределами графика), легенда опять становится видимой. А двойной щелчок на линии последовательности с выделенными точками изменит цвет линии. Если выделена только одна точка последовательности, включается или отключается метка данных возле этой точки:

```
Private Sub Chart_BeforeDoubleClick(ByVal ElementID As Long, _
  ByVal Arg1 As Long, ByVal Arg2 As Long, Cancel As Boolean)

  Dim theSeries As Series

  Select Case ElementID
    Case xlLegend
      Me.HasLegend = False
      Cancel = True
    Case xlChartArea
```

```
      Me.HasLegend = True
      Cancel = True
    Case xlSeries
      'Arg1 содержит индекс последовательности
      'Arg2 содержит индекс точек (-1 если выделена вся
      'последовательность)
      Set theSeries = Me.SeriesCollection(Arg1)
  If Arg2 = -1 Then
        With theSeries.Border
          If .ColorIndex = xlColorIndexAutomatic Then
            .ColorIndex = 1
          Else
            .ColorIndex = (.ColorIndex Mod 56) + 1
          End If
            End With
          Else
            With theSeries.Points(Arg2)
              .HasDataLabel = Not .HasDataLabel
            End With
        End If
        Cancel = True
  End Select
End Sub
```

С помощью параметра `ElementID` передается идентификатор элемента, на котором пользователь выполнил двойной щелчок. Для определения элемента можно применять встроенные константы, например `xlLegend`. В завершение каждого блока параметру `Cancel` присваивается значение `True`. Это позволяет отменить принятый по умолчанию обработчик события двойного щелчка. В таком случае окно с параметрами форматирования не выводится.

Обратите внимание, как ключевое слово `Me` используется в качестве ссылки на объект, связанный с модулем кода. Применение `Me` вместо `Chart1` позволяет сделать код переносимым и подходящим для использования вместе с другими диаграммами. На самом деле, можно даже отказаться от ссылки на объект `Me` и использовать `HasLegend =`. В модуле класса соответствующего объекта на свойства этого же объекта можно ссылаться без квалификации. Но квалификация свойств позволяет отличать свойства объекта от переменных.

Если элемент диаграммы является последовательностью, то в параметре `Arg1` содержится индекс последовательности в коллекции `SeriesCollection`. Если выделена единственная точка в последовательности, то в параметре `Arg2` передается индекс точки, а если выделена вся последовательность, параметр `Arg2` имеет значение 1.

При выделении всей последовательности процедура обработки события использует значение 1 для установки индекса цвета границы последовательности (если применяется автоматический индекс цвета). Если автоматический индекс не используется, процедура увеличивает индекс цвета на 1. Так как доступно только 56 цветов, перед добавлением 1 процедура применяет оператор `Mod`, который делит индекс цвета на 56 и возвращает остаток. Эта операция влияет только на значение индекса 56. Выражение `56 Mod 56` возвращает ноль, а это значит, что следующим после 56 индексом цвета будет 1.

При выборе единственной точки последовательности процедура переключает состояние метки данных этой точки. Если свойство `HasDataLabel` этой точки установлено в значение `True`, оператор `Not` превращает его в `False`. Если свойство `HasDataLabel` установлено в значение `False`, оператор `Not` превращает его в `True`.

События книги

Возможны следующие процедуры обработки событий книги. Процедуры, появившиеся с выходом Excel 2003, выделены полужирным шрифтом.

- Private Sub Workbook_Activate()
- Private Sub Workbook_AddinInstall()
- Private Sub Workbook_AddinUninstall()
- **Private Sub Workbook_AfterXmlExport(ByVal Map As XmlMap, ByVal Url As String, ByVal Result As XlXmlExportResult)**
- **Private Sub Workbook_AfterXmlImport(ByVal Map As XmlMap, ByVal IsRefresh As Boolean, ByVal Result As XlXmlImportResult)**
- Private Sub Workbook_BeforeClose(Cancel As Boolean)
- Private Sub Workbook_BeforePrint(Cancel As Boolean)
- Private Sub Workbook_BeforeSave(ByVal SaveAsUI As Boolean, Cancel As Boolean)
- **Private Sub Workbook_BeforeXmlExport(ByVal Map As XmlMap, ByVal Url As String, Cancel As Boolean)**
- **Private Sub Workbook_BeforeXmlImport(ByVal Map As XmlMap, ByVal Url As String, ByVal IsRefresh As Boolean, Cancel As Boolean)**
- Private Sub Workbook_Deactivate()
- Private Sub Workbook_NewSheet(ByVal Sh As Object)
- Private Sub Workbook_Open()
- Private Sub Workbook_PivotTableCloseConnection(ByVal Target As PivotTable)
- Private Sub Workbook_PivotTableOpenConnection(ByVal Target As PivotTable)
- Private Sub Workbook_SheetActivate(ByVal Sh As Object)
- Private Sub Workbook_SheetBeforeDoubleClick(ByVal Sh As Object, ByVal Target As Range, Cancel As Boolean)
- Private Sub Workbook_SheetBeforeRightClick(ByVal Sh As Object, ByVal Target As Range, Cancel As Boolean)
- Private Sub Workbook_SheetCalculate(ByVal Sh As Object)
- Private Sub Workbook_SheetChange(ByVal Sh As Object, ByVal Target As Range)
- Private Sub Workbook_SheetDeactivate(ByVal Sh As Object)
- Private Sub Workbook_SheetFollowHyperlink(ByVal Sh As Object, ByVal Target As Hyperlink)
- Private Sub Workbook_SheetPivotTableUpdate(ByVal Sh As Object, ByVal Target As PivotTable)
- Private Sub Workbook_SheetSelectionChange(ByVal Sh As Object, ByVal Target As Range)
- **Private Sub Workbook_Sync(ByVal SyncEventType As Office.MsoSyncEventType)**
- Private Sub Workbook_WindowActivate(ByVal Wn As Window)

- Private Sub Workbook_WindowDeactivate(ByVal Wn As Window)
- Private Sub Workbook_WindowResize(ByVal Wn As Window)

Кое-какие из процедур обработки событий позволяют обрабатывать события листов и диаграмм, но на уровне книги. Разница заключается в том, что при создании такой процедуры (например, для обработки события Change) в пределах листа или диаграммы, процедура срабатывает только в пределах листа. При создании процедуры обработки события на уровне книги (например, для обработки события SheetChange) процедура срабатывает на любом листе книги.

Чаще всего на уровне книги используется процедура обработки события Open, в которой выполняется инициализация книги при открытии. Здесь можно настроить режим расчета, указать параметры экрана, изменить структуру меню, выбрать расположение панелей инструментов или ввести данные в комбинированные списки или списки на листах.

Точно так же процедура обработки события Workbook_BeforeClose может использоваться для очистки среды перед закрытием книги. Например, эта процедура применяется для восстановления параметров экрана и содержимого меню, кроме этого, для предотвращения закрытия книги. Для этого параметр Cancel должен устанавливаться в значение True. В следующей процедуре обработки событий закрытие книги разрешается только в том случае, если значение ячейки FinalProfit находится в пределах от 500 до 600.

```
Private Sub Workbook_BeforeClose(Cancel As Boolean)
  Dim Profit As Double

  Profit = ThisWorkbook.Worksheets(2).Range("FinalProfit").Value
  If Profit < 500 Or Profit > 600 Then
    MsgBox "Прибыль должна находится в диапазоне от 500 до 600"
    Cancel = True
  End If
End Sub
```

Обратите внимание, что, если в процедуре обработки события BeforeClose параметру Cancel присваивается значение True, Excel также не сможет завершить работу.

Сохранение изменений

Если необходимо обеспечить сохранение всех изменений при закрытии книги, но пользователь не должен получать запрос на сохранение, сохранение можно запускать в процедуре обработки события BeforeClose. Необходимость сохранения определяется значением свойства Saved объекта книги. Если в книге присутствуют несохраненные изменения, свойство устанавливается в значение False.

```
Private Sub Workbook_BeforeClose(Cancel As Boolean)
   If Not ThisWorkbook.Saved Then
      ThisWorkbook.Save
   End If
End Sub
```

С другой стороны, если необходимо отказаться от сохранения изменений в книге и запретить выдачу пользователю запроса в момент закрытия, можно установить свойство Saved в значение True. Это также можно сделать в процедуре обработки события BeforeClose.

```
Private Sub Workbook_BeforeClose(Cancel As Boolean)
   ThisWorkbook.Saved = True
End Sub
```

После этого Excel будет считать, что все изменения сохранены.

Верхние и нижние колонтитулы

Часто в Excel возникает необходимость печати информации в верхнем или нижнем колонтитуле страницы. Иногда эта информация извлекается из ячейки на листе или не поддерживается доступными параметрами колонтитулов. Например, можно вывести название компании, хранящееся в ячейке на листе, или вывести полный путь к файлу книги.

Полный путь и имя файла доступны как один из вариантов содержимого колонтитулов в Excel 2003. Но для добавления в колонтитул текста из ячейки листа потребуется использование кода. В следующей процедуре текст из ячейки A2 листа Profit размещается в левой части нижнего колонтитула, после чего очищается центр нижнего колонтитула и в правой части колонтитула указывается полный путь к имени файла. Эти изменения вносятся на каждом листе и диаграмме в пределах книги:

```
Private Sub Workbook_BeforePrint(Cancel As Boolean)
   Dim aWorksheet As Worksheet
   Dim FullFileName As String
   Dim CompanyName As String

   CompanyName = Worksheets("Profit").Range("A2").Value

   FullFileName = ThisWorkbook.FullName
  For Each aWorksheet In ThisWorkbook.Worksheets
    With aWorksheet.PageSetup
      .LeftFooter = CompanyName
      .CenterFooter = ""
      .RightFooter = FullFileName
    End With
  Next aWorksheet

  Dim aChart As Chart

  For Each aChart In ThisWorkbook.Charts
    With aChart.PageSetup
      .LeftFooter = CompanyName
      .CenterFooter = ""
      .RightFooter = FullFileName
    End With
  Next aChart

End Sub
```

Содержимое нижнего колонтитула можно просмотреть в режиме предварительного просмотра (рис. 5.3)

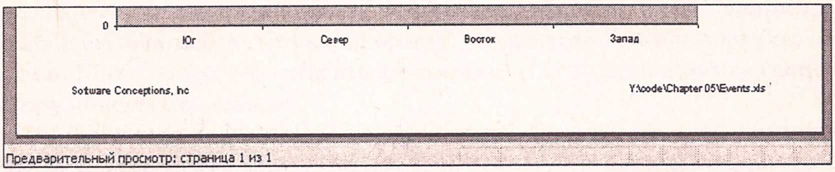


Рис. 5.3. Нижний колонтитул диаграммы в режиме предварительного просмотра

Резюме

В этой главе были рассмотрены полезные примеры использования процедур обработки событий в ответ на действия пользователей.

Речь шла о событиях книг, листов и диаграмм. Более подробно проанализированы следующие события:

- Worksheet_Calculate
- Chart_BeforeDoubleClick
- Workbook_BeforeClose
- Workbook_BeforePrint

Язык VBA на самом деле является языком программирования на основе событий. Следовательно, знакомство с ними позволяет получить доступ к новой функциональности, о которой многие даже не подозревают.

Дополнительную информацию можно найти в окне Object Browser (Просмотр объектов) и в приложении А.

6

Модули классов

Модули классов используются в VBA для создания собственных классов. Вот несколько примеров проблем, решение которых требует создания собственных модулей классов:

- реакция на события приложения; например, можно создавать код, который будет выполняться при сохранении или печати любой книги;
- реакция на события встроенных диаграмм;
- создание единой процедуры обработки событий, которая будет использоваться большим количеством элементов управления ActiveX, например текстовыми полями в диалоговом окне `UserForm`;
- инкапсуляция методов Windows API для взаимодействия с операционной системой Windows;
- инкапсуляция стандартных процедур VBA в удобной для переноса в другие книги форме.

В этой главе будут определены некоторые классы, позволяющие понять, как работают модули классов. После этого изученные принципы можно использовать в более полезных примерах. Ранее уже расматривались встроенные объекты Excel, например объект `Worksheet`, и было показано, что часто объекты являются частью коллекции, например, `Worksheets`. Кроме этого, объекты имеют свойства и методы, например свойство `Name` и метод `Copy` объекта `Worksheet`.

Модуль класса позволяет создать собственную "схему" объекта, например класс `Employee`. В модуле класса можно определить свойства и методы, например свойство `Rate`, в котором записывается текущая заработная плата работника, и метод `ChangeTitle`, неявно затрагивающий другие фрагменты состояния объекта `Employee`. Кроме этого, можно создать новую коллекцию, например `Employees`. Модуль класса является описа-

нием объектов, которые должны создаваться. С его помощью создаются экземпляры класса. Например, Mary, Jack и Anne могут быть экземплярами класса Employee и входить в коллекцию Employees.

Создание собственных объектов

Рассмотрим процесс создания объекта Employee, описанного выше. В объекте необходимо хранить имя работника, количество рабочих часов в неделю и размер оплаты труда. Объект Employee можно создавать с тремя свойствами, которые будут хранить необходимые данные, и методом, рассчитывающим размер еженедельной оплаты.

Для этого создается модуль класса, называемый Employee. Он показан в верхней правой части рис. 6.1.

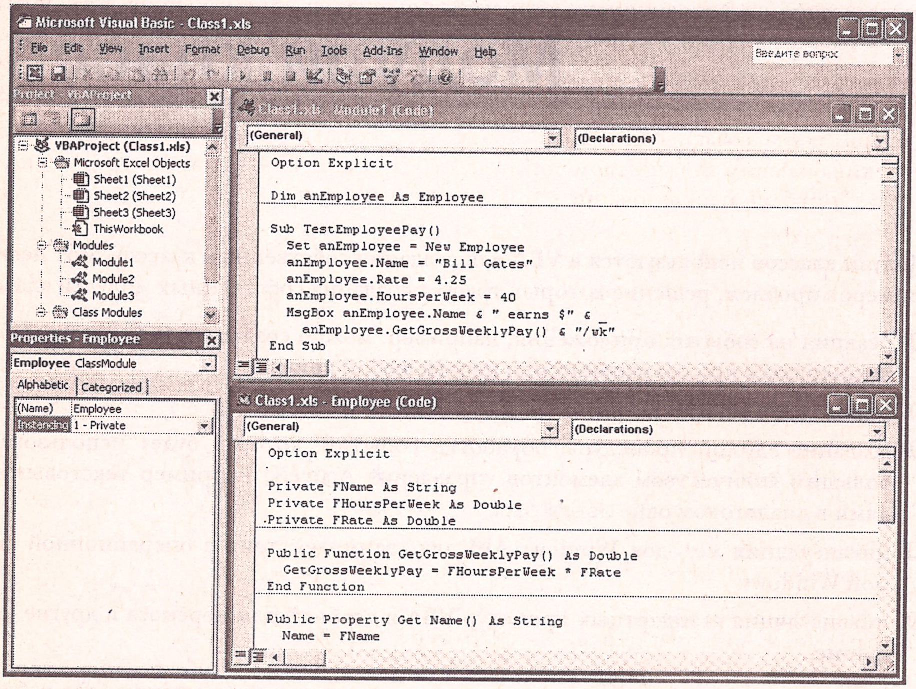


Рис. 6.1. Определение и тестирование класса в редакторе VBE

В модуле класса Employee объявляется три открытых свойства Name, HoursPerWeek и Rate (на рис. 6.1 не показано). Кроме этого, предоставляется один открытый метод GetGrossWeeklyPay.

Код в стандартном модуле (в нижней правой части рис. 6.1) создает объект Employee на основе схемы, описанной в модуле класса Employee. В модуле объявляется переменная anEmployee, имеющая тип Employee. Подпрограмма TestEmployeePay использует оператор Set для присваивания нового экземпляра класса Employee переменной anEmployee.

После этого подпрограмма присваивает значения трем свойствам объекта и генерирует сообщение, которое выводится в окне сообщения. Для формирования сообще-

ния используется значение свойства Name объекта Employee и вызывается метод GetGrossWeeklyPay.

Ниже показан альтернативный вариант стандартного модуля кода, который стоит применить, когда необходим только один экземпляр класса:

```
Dim anEmployee As New Employee

Sub EmployeePay()
   anEmployee.Name = "Mary"
   anEmployee.Rate = 15
   anEmployee.HoursPerWeek = 35
   MsgBox anEmployee.Name & " зарабатывает $" _
   & anEmployee.GetGrossWeeklyPay() & " в неделю"
End Sub
```

В данном случае в строке объявления используется ключевое слово New. При этом объект класса Employee создается автоматически при первом же применении в коде модуля.

Использование коллекций

Теперь, когда создан один класс Employee, пришло время создать механизм для управления большим количеством объектов Employee. В VBA для этого предоставляется класс Collection, использование которого показано ниже:

```
Option Explicit
Dim Employees As New Collection

Sub TestEmployeesCollection()
  Dim anEmployee As Employee

  Dim I As Long

  ' Удалить сотрудников из коллекции
  For I = Employees.Count To 1 Step -1
  Call Employees.Remove(I)
Next I

Set anEmployee = New Employee
anEmployee.Name = "Paul Kimmel"
anEmployee.Rate = 15
anEmployee.HoursPerWeek = 45
Call Employees.Add(anEmployee, anEmployee.Name)

Set anEmployee = New Employee
anEmployee.Name = "Bill Gates"
anEmployee.Rate = 14
anEmployee.HoursPerWeek = 35
Call Employees.Add(anEmployee, anEmployee.Name)

MsgBox "Количество сотрудников составляет " & Employees.Count
MsgBox "Employees(2).Name = " & Employees(2).Name
MsgBox "Employees(""Paul Kimmel"").Rate = " _
  & Employees("Paul Kimmel").Rate

For Each anEmployee In Employees
  MsgBox anEmployee.Name & " зарабатывает $" _
    & anEmployee.GetGrossWeeklyPay()
Next anEmployee
End Sub
```

В начале стандартного модуля переменная Employees объявляется как объект класса Collection. Процедура TestEmployeeCollection использует метод Remove объекта коллекции в цикле For... Next для удаления существующих объектов в обратном порядке, так как удаление в прямом порядке приведет к изменению верхней границы, от которой зависит цикл. Обычно этот шаг не нужен, так как сразу после создания коллекции не содержат объектов. Код используется только для демонстрации метода Remove и для поддержки многократного запуска процедуры без удвоения количества объектов в коллекции.

Процедура TestEmployeeCollection создает первого работника, "Paul Kimmel", и использует метод Add для добавления объекта Employee в коллекцию. Первый параметр метода Add содержит ссылку на объект. Второй, необязательный, параметр содержит идентификатор, по которому на объект можно будет ссылаться в дальнейшем. В данном случае используется свойство Name коллекции Employees. Эта процедура применяется и для добавления второго работника.

> Если каждому члену коллекции предоставляется ключ, ключи должны быть уникальны. Если в коллекцию добавить объект, ключ которого совпадает с уже использующимся ключом, появится сообщение об ошибке времени выполнения. Не рекомендуется использовать в качестве ключа имя работника, так как люди могут иметь одинаковые имена. Для создания ключа стоит применять уникальные идентификаторы, например номера социального страхования (хотя практика показывает, что с точки зрения безопасности такая информация является не лучшим выбором уникального ключа. — *Примеч. ред.*)

Операторы MsgBox показывают, что на коллекцию можно ссылаться так же, как на встроенные коллекции Excel. Например, коллекция Employees имеет свойство Count. На членов коллекции можно ссылаться по позиции или по ключу, если объектам были предоставлены значения ключей.

Коллекция в модуле класса

Коллекцию можно создать и в модуле класса. Этот подход имеет свои преимущества и недостатки. Преимуществом является увеличение контроля над взаимодействием с коллекцией. При этом можно ограничить непосредственный доступ к коллекции и инкапсулировать код в пределах одного модуля, что позволит сделать код более переносимым и более простым в сопровождении. Недостатком этого подхода является большая трудоемкость создания. Кроме этого, становятся недоступны некоторые простые способы обращения как к элементам коллекции, так и к самой коллекции.

В следующем коде показано содержимое модуля класса Employees:

```
Option Explicit

Private FEmployees As New Collection

Public Function Add(ByVal value As Employee)
  Call FEmployees.Add(value, value.Name)
End Function

Public Property Get Count() As Long
  Count = FEmployees.Count
End Property

Public Property Get Items() As Collection
  Set Items = FEmployees
```

```
End Property

Public Property Get Item(ByVal value As Variant) As Employee
  Set Item = FEmployees(value)
End Property

Public Sub Remove(ByVal value As Variant)
  Call FEmployees.Remove(value)
End Sub
```

Если коллекция выделена в отдельный модуль класса, обращение к четырем методам коллекции (Add, Count, Item и Remove) из стандартного модуля оказывается невозможным. В модуле класса придется создавать собственные методы и свойства, даже если модификация методов коллекции не требуется. С другой стороны, разработчик может выбирать реализуемый фрагмент функциональности, а также, что предоставлять в виде метода, а что — в виде свойства.

В классе Employees функция Add, подпрограмма Remove, свойство Get Item и свойство Count реализуют большую часть функциональности коллекции. В процедуре свойства Get Items присутствует одна новая возможность. В то время как свойство Get Item возвращает ссылку на один элемент коллекции, свойство Get Items возвращает ссылку на всю коллекцию. Эта возможность позволяет использовать коллекцию в циклах For Each... Next.

При этом стандартный модуль кода будет выглядеть следующим образом:

```
Option Explicit
Dim theEmployees As New Employees

Sub TestEmployeesCollection()
Dim anEmployee As Employee

Dim I As Long

' Удалить сотрудников из коллекции
For I = theEmployees.Count To 1 Step -1
  Call theEmployees.Remove(I)
Next I

Set anEmployee = New Employee
anEmployee.Name = "Paul Kimmel"
anEmployee.Rate = 15
anEmployee.HoursPerWeek = 45
Call theEmployees.Add(anEmployee)

Set anEmployee = New Employee
anEmployee.Name = "Bill Gates"
anEmployee.Rate = 14
anEmployee.HoursPerWeek = 35
Call theEmployees.Add(anEmployee)

MsgBox "Количество сотрудников = " & theEmployees.Count
MsgBox "Employees.Item(2).Name = " & theEmployees.Item(2).Name
MsgBox "Employees.Item(""Paul Kimmel"").Rate = " _
  & theEmployees.Item("Paul Kimmel").Rate

For Each anEmployee In theEmployees.Items
  MsgBox anEmployee.Name & " зарабатывает $" _
    & anEmployee.GetGrossWeeklyPay()
Next anEmployee
End Sub
```

Объектная переменная theEmployees объявлена как экземпляр класса Employees. Как и раньше, из коллекции удаляются объекты, и цикл For... Next добавляет в коллекцию двух работников. Одним из дополнительных удобств является отсутствие необходимости добавлять значение ключа при использовании метода Add коллекции Employees. Вместо разработчика этим занимается код метода Add.

Первый, второй, третий и четвертый операторы MsgBox показывают новые свойства, которые необходимы для получения ссылок на коллекцию и элементы коллекции. Для получения ссылки на элемент коллекции нужно использовать свойство Item, а для получения ссылки на всю коллекцию — свойство Items.

Перехват событий приложения

Модуль класса можно использовать для перехвата событий приложения. Большинство этих событий совпадают с событиями книги, но относятся ко всем открытым книгам, а не только к книге, содержащей процедуру обработки события. Например, в книге существует событие BeforePrint, которое возникает перед печатью любого фрагмента книги. На уровне приложения есть событие WorkbookBeforePrint, возникающее перед запуском печати в любой книге.

Для получения списка доступных событий приложения необходимо добавить в проект модуль класса, который может иметь любое допустимое имя. На следующем снимке экрана показан модуль класса AppEvents. После этого в начало модуля можно ввести следующее определение переменной:

```
Public WithEvents App As Application
```

Вместо App используется любое допустимое имя объектной переменной. В коде, ссылающемся на модуль класса, это имя применяется в качестве имени свойства класса. Ключевое слово WithEvents позволяет получать доступ к событиям, которые связаны с объектом приложения. Теперь App можно выбрать из левого раскрывающегося списка в верхней части модуля. После этого в правом раскрывающемся списке будут перечислены соответствующие события (рис. 6.2).

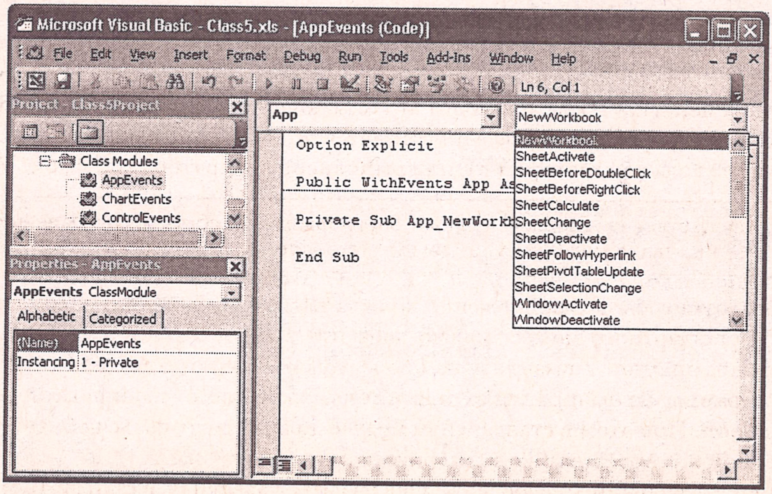


Рис. 6.2. Выбор событий класса

В данном случае выбирается событие WorkbookBeforePrint и расширяется процедура обработки события, которая описывалась в главе, посвященной событиям. Для этого в модуле класса AppEvents используется следующий код:

```
Private Sub App_WorkbookBeforePrint(ByVal Wb As Workbook, _
  Cancel As Boolean)

  Dim aWorksheet As Worksheet
  Dim FullFileName As String
  Dim CompanyName As String

  With Wb
    CompanyName = "Software Conceptions, Inc"
    FullFileName = .FullName
    For Each aWorksheet In .Worksheets
      With aWorksheet.PageSetup
        .LeftFooter = CompanyName
        .CenterFooter = ""
        .RightFooter = FullFileName
      End With
    Next aWorksheet
  End With
End Sub
```

В отличие от модулей классов книг и листов, процедуры обработки событий, добавленные в собственные модули классов, не работают по умолчанию. Для этого придется создать экземпляр класса и присвоить объект Application свойству App созданного объекта. В стандартном модуле должен присутствовать следующий код:

```
Public theAppEvents As New AppEvents

Sub TrapApplicationEvents()
   Set theAppEvents.App = Application
End Sub
```

После этого достаточно вызвать процедуру TrapApplicationEvents. В результате процедура обработки события WorkbookBeforePrint будет запускаться при каждом использовании команд Печать (Print) или Предварительный просмотр (Preview). Процедура будет запускаться до закрытия книги, содержащей процедуру обработки события.

Существует возможность отключения перехвата событий приложения без завершения текущего сеанса. Любое действие, приводящее к сбросу переменных уровня модуля или открытых переменных, также приводит и к отключению обработки событий приложения, так как экземпляр класса также уничтожается. К сбросу переменных может привести редактирование кода в редакторе VBE или выполнение оператора End в коде VBA.

> В Excel 97 существовали редкие ошибки, приводящие к сбросу переменных. Можно предполагать, что такие ошибки существуют и в более поздних версиях Excel.

Если обработка событий приложения должна работать для всех сеансов Excel, модуль класса и код стандартного модуля можно добавить в книгу Personal.xls, а процедуру TrapApplicationEvents вызывать из процедуры обработки события Workbook_Open. Код подпрограммы TrapApplicationEvents можно даже скопировать в процедуру Workbook_Open. При этом в стандартном модуле должно сохраняться объявление объектной переменной theAppEvents с квалификатором доступа Public.

Например, следующий код добавляется в раздел объявлений в стандартном модуле:

```
Public theAppEvents As New AppEvents
```

При этом в модуль ThisWorkbook можно добавить следующую процедуру обработки события:

```
Private Sub Workbook_Open()
    Set theAppEvents.App = Application
End Sub
```

Встроенные события диаграмм

Если необходимо перехватывать события для встроенных в лист диаграмм, воспользуйтесь способом, применяемым для перехвата событий приложения. Сначала необходимо в проекте создать новый модуль класса (можно использовать тот же модуль класса, который применялся для перехвата событий приложения). В начало модуля класса необходимо вставить следующее определение:

```
Public WithEvents aChart As Chart
```

Создадим процедуру обработки события BeforeDoubleClick, которая использовалась в главе 5. Модуль класса должен выглядеть следующим образом:

```
Public WithEvents aChart As Chart
```

```
Private Sub aChart_BeforeDoubleClick(ByVal ElementID As Long, _
  ByVal Arg1 As Long, ByVal Arg2 As Long, Cancel As Boolean)

  Dim theSeries As Series

  Select Case ElementID
    Case xlLegend
      aChart.HasLegend = False
      Cancel = True
    Case xlChartArea
      aChart.HasLegend = True
      Cancel = True
    Case xlSeries
      'Arg1 является индексом последовательностей
      'Arg2 является индексом точек (-1 если выделена вся
      'последовательность)
      Set theSeries = aChart.SeriesCollection(Arg1)
      If Arg2 = -1 Then
        With theSeries.Border
          If .ColorIndex = xlColorIndexAutomatic Then
            .ColorIndex = 1
          Else
            .ColorIndex = (.ColorIndex Mod 56) + 1
          End If
              End With
          Else
            With theSeries.Points(Arg2)
              .HasDataLabel = Not .HasDataLabel
            End With
        End If
        Cancel = True
  End Select
End Sub
```

Этот код позволяет удалять легенду диаграммы с помощью двойного щелчка. Двойной щелчок на области диаграммы приводит к появлению легенды. Если выполнить двойной щелчок на линии последовательности, она изменит цвет. Если выделить точку в последовательности и выполнить на точке двойной щелчок, возле точки появится или исчезнет метка данных.

Предположим, диаграмма хранится в объекте ChartObject, который является единственным объектом этого класса на листе Манго. При этом модуль класса называется ChartEvents. В стандартный модуль необходимо добавить следующий код:

```
Public theChartEvents As New ChartEvents

Sub InitializeChartEvents()
  Set theChartEvents.aChart = _
    ThisWorkbook.Worksheets("Манго").ChartObjects(1).Chart
End Sub
```

Полученная в результате диаграмма показана на рис. 6.3.

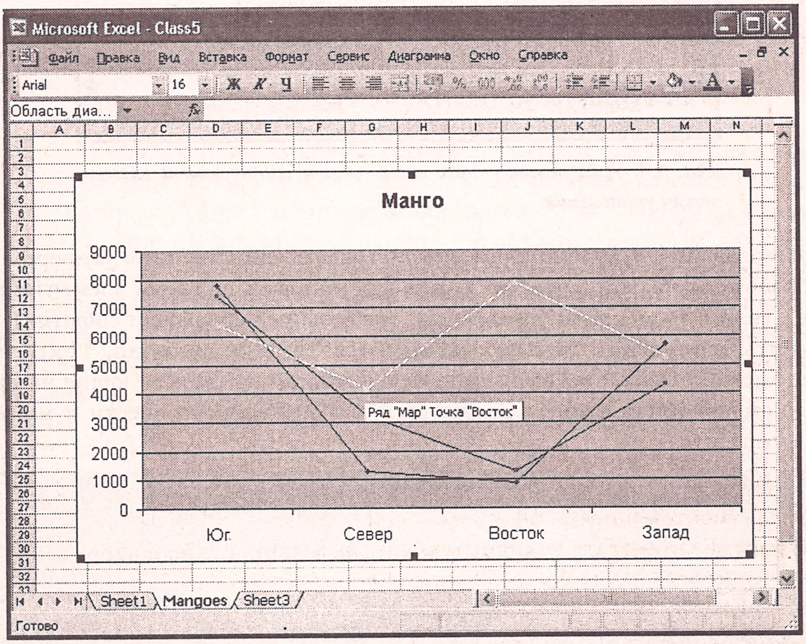


Рис. 6.3. Перехват событий диаграммы

После выполнения процдеры InitializeChartEvents двойной щелчок на последовательности, точке или легенде приводит к запуску процедуры обработки события BeforeDoubleClick.

Коллекция элементов управления UserForm

Если на форме находится несколько элементов управления одного типа, для каждого из них обычно создаются практически идентичные процедуры обработки событий. Например, если двойной щелчок на метке слева от поля ввода TextBox должен приводить

к очистке поля и установке на нем фокуса, потребуется создание четырех практически идентичных процедур обработки событий, по одной для каждого элемента управления, как показано на рис. 6.4.

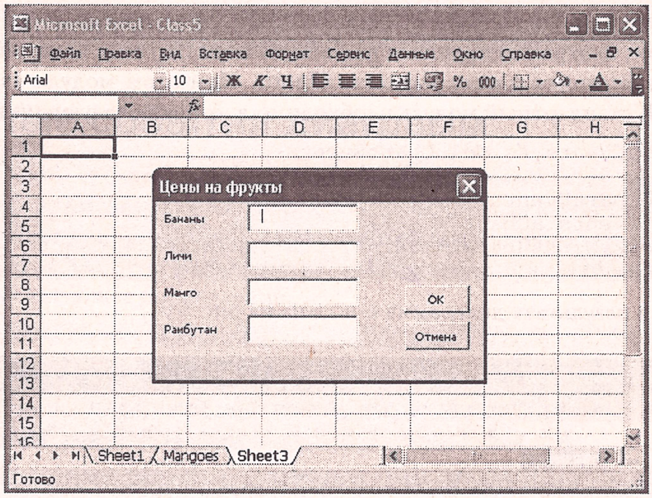


Рис. 6.4. Диалоговое окно с несколькими одинаковыми элементами управления

Воспользовавшись модулем класса, можно создать единственную универсальную процедуру обработки событий, которую можно использовать со всеми элементами управления `Label` (или с элементами управления, требующими обработки события). Для удобства элементы управления `TextBox` называются `TextBoxBananas`, `TextBoxLyches`, `TextBoxangoes` и `TextBoxRambutan`. Метки имеют соответствующие названия с использованием префикса `Label`. Следующий код необходимо ввести в модуль класса `ControlEvents`:

```
Public WithEvents Label As MSForms.Label
Public Form As UserForm

Private Sub Label_DblClick(ByVal Cancel As MSForms.ReturnBoolean)
  Dim Product As String
  Dim TextBoxName As String
Product = Mid(Label.Name, 6)
  TextBoxName = "TextBox" & Product
  With Form.Controls(TextBoxName)
    .Text = ""
    .SetFocus
  End With
End Sub
```

Объект `Label` объявляется с событиями как метка диалогового окна `UserForm`. Объект `Form` объявляется как диалоговое окно `UserForm`. Универсальная процедура обработки события для объекта `Label` называется `DblClick` и использует функцию `Mid` для получения названия товара начиная с шестого символа в имени метки (после идентификатора `"Label"`). Имя объекта `TextBox` получается с помощью добавления префикса `"TextBox"` к названию продукта.

Структура With... End With идентифицирует объект TextBox, используя имя TextBox в качестве индекса в коллекции Controls диалогового окна UserForm. Свойству Text объекта TextBox присваивается строка нулевой длины и с помощью метода SetFocus курсор устанавливается в поле ввода.

Следующий код вводится в модуль класса диалогового окна UserForm:

```
Option Explicit

Dim Labels As New Collection

Private Sub UserForm_Initialize()
  Dim Control As MSForms.Control
  Dim aControlEvent As ControlEvents

  For Each Control In Me.Controls
    If TypeOf Control Is MSForms.Label Then
      Set aControlEvent = New ControlEvents
      Set aControlEvent.Label = Control
      Set aControlEvent.Form = Me
      Call Labels.Add(aControlEvent)
    End If
  Next Control
End Sub
```

Labels — это новая коллекция, в которой хранятся объекты, созданные в модуле класса ControlEvents. В процедуре обработки события Initialize диалогового окна UserForm элементы управления меток связываются с экземплярами класса ControlEvents.

Цикл For Each ... Next по очереди обрабатывает все элементы управления из диалогового окна. При обнаружении элемента управления, который является меткой (для определения типа элемента управления используется ключевое слово TypeOf) создается новый экземпляр класса ControlEvents и полученный экземпляр присваивается объектной переменной aControlEvent. Свойству Label созданного объекта присваивается ссылка на элемент управления, а свойству Form — ссылка на диалоговое окно. После этого созданный объект добавляется в коллекцию Labels.

При загрузке в память диалогового окна UserForm запускается процедура обработки события Initialize. После запуска процедура связывает метки с процедурой обработки событий из модуля класса. Двойной щелчок на любой метке приводит к очистке содержимого поля ввода (элемент управления TextBox), которое находится справа от метки. После двойного щелчка на метке и очистки содержимого в поле ввода можно вводить новые данные.

> Стоит обратить внимание, что ряд событий, связанных с некоторыми элементами управления, не доступны в модуле класса при использовании оператора With Events. Например, самыми полезными событиями полей ввода на диалоговых окнах UserForm являются BeforeUpdate, AfterUpdate, Enter и Exit. Ни одно из этих событий не доступно в модуле класса. Эти события можно обрабатывать только в модуле класса, связанном с диалоговым окном UserForm.

Ссылки на классы из других проектов

Если макрос необходимо использовать в другой книге, выберите пункт Tools⇨References (Сервис⇨Ссылки) в окне редактора VBE и создайте ссылку на проект VBA, связанный с другой книгой. Ссылка отображается как специальный объект в окне Project Explorer (Окно проекта) (рис. 6.5).

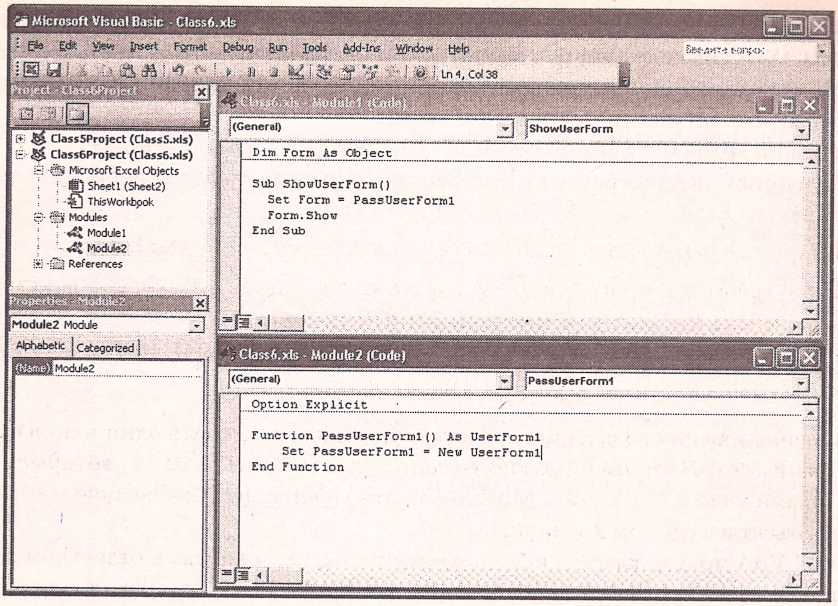


Рис. 6.5. Ссылка в окне Project Explorer

В проекте `Class6.xls` присутствует ссылка на проект `Class5.xls`. В проекте `Class5.xls` присутствует диалоговое окно `UserForm` из предыдущего примера. Ссылка позволяет запускать в стандартных модулях книги `Class6.xls` процедуры из стандартных модулей книги `Class5.xls`. При этом ссылка не дает возможности создавать экземпляры классов или диалоговых окон `UserForm` из книги, на которую указывает ссылка.

> При создании ссылки на другую книгу необходимо убедиться, что проект VBA в интересующей книге имеет уникальное имя. По умолчанию проект называется `VBAProject`. Выберите пункт Tools⇨VBA Project Properties (Сервис⇨Свойства проекта) и введите новое имя проекта.

Для обхода ограничения на использование диалоговых окон `UserForm` и классов из целевой книги в ней можно создать функцию, которая возвращает ссылку на диалоговое окно. Пример такой функции показан в правой нижней части последнего снимка экрана. Функция `PassUserForm1` из книги `Class5.xls` возвращает новый экземпляр класса `UserForm1` в качестве собственного значения. В книге `Class6.xls` переменная `Form` объявляется как имеющая универсальный тип `Object`. Процедура `ShoUserForm` присваивает возвращаемое значение функции `PassUserForm1` переменной `Form`. После этого переменную `Form` можно использовать для вывода диалогового окна `UserForm` и получения доступа к элементам управления. При этом диалоговое окно может быть скрыто, но не должно выгружаться из памяти.

Резюме

Модули классов применяются для создания схем новых объектов. В качестве примера такой схемы в этой главе использовался класс Employee.

- Функции и подпрограммы в модуле класса применялись для определения методов объекта.
- Открытые переменные использовались для определения свойств объекта.
- Если требуется программное управление значениями свойств, свойство можно определить с помощью процедуры Property Let.
- Кроме этого, процедуры Property Get позволяют управлять доступом к значениям свойств.

Для использования кода из модуля класса необходимо создать один или несколько экземпляров класса. Например, можно создать объекты Paul и Bill, которые являются экземплярами класса Employee. Можно создать собственную коллекцию, позволяющую хранить объекты в едином хранилище.

В Excel VBA модули классов используются не так широко, как в отдельном языке программирования Visual Basic. Это связано с тем, что Excel и так предоставляет достаточное количество объектов, доступных для разработчиков. При этом разработчики приложений для Excel могут использовать модули классов для решения таких задач:

- обработка событий уровня приложения, например WorkbookBeforePrint, которое позволяет управлять печатью всех открытых книг;
- обработка событий для встроенных диаграмм;
- создание единственной процедуры обработки событий, которая может использоваться несколькими экземплярами определенного класса, например элементами управления TextBox на диалоговом окне UserForm;
- инкапсуляция сложного кода и упрощение его применения;
- инкапсуляция кода для повторного использования;
- дифференцирование проектов и пользователей.

Дополнительная информация об инкапсуляции кода API приводится в главе 16.

7

Создание надежного кода

Надежный код можно сравнить с Суперменом и бронежилетом. Супермен неуязвим (если рядом нет криптонита). Бронежилет также делает владельца неуязвимым, пока кто-то не воспользуется бронебойными пулями (или, что интересно, композитным луком). Иногда неуязвимость рассматривается как что-то нереальное и невозможное, а иногда неуязвимость вполне возможна, но с определенными ограничениями.

Существующая технология не позволяет создавать алгоритмы, которые математически доказывают эффективность приложения, поэтому невозможно доказать, что оно совершенно лишено ошибок или неуязвимо. Все считают, что Супермен сделан из стали и непобедим, а представители силовых структур просто носят бронежилеты. Разработчики также применяют средства защиты при создании кода приложений. В этой главе рассматриваются методики, обеспечивающие жизнеспособность приложения и позволяющие защитить его от аварийного завершения работы, удаления файлов или более нежелательных вариантов поведения. Если что-то пойдет не так, можно будет воспользоваться описанными здесь приемами диагностики и решения проблемы. Кроме этого, описанные подходы являются переносимыми и могут применяться при создании различных приложений.

Обсуждение основано на проверенной десятилетиями стратегии создания защищенного кода с помощью инструментов, предоставленных интерпретатором VBA. Здесь рассматриваются способы обнаружения неоправдавшихся предположений и методика контроля за порядком передачи управления в пределах кода, что позволяет оставлять "хлебные крошки" и проверять каждый путь, ветвление и цикл кода. В результате применения этих методик остается меньше потенциальных проблем. Если проблемы все же возникают, разработчик может воспользоваться доступной информацией для их быстрой диагностики и решения.

Использование метода Debug.Print

Объект Debug предоставляет несколько методов. Краеугольным камнем фундамента инструментов отладки и тестирования является метод Debug.Print. Еще один метод, Debug.Assert, будет рассматриваться в следующем разделе.

Debug.Print — очень простой метод. Он принимает строку с сообщением и отправляет сообщение в окно Immediate (Проверка). Кроме этого, метод Debug.Print выполняет свою задачу только при запуске кода в режиме отладки (в редакторе VBE), что делает его идеальным инструментом для отслеживания реального текущего состояния кода, а не предполагаемого.

Как в свое время сказал Рональд Рейган: “Доверяй, но проверяй”. Мы, как разумные программисты, верим, что код будет выполняться в том порядке, который *имел в виду* программист, но на самом деле код всегда выполняется в том порядке, который *был записан* программистом. Метод Debug.Print позволяет точно узнать, как выполняется код.

Создадим инструментарий отладки, воспользовавшись в качестве основы методом Debug.Print. Вызовы Debug.Print можно вставлять в любой метод или свойство, например:

```
Debug.Print "Шляпу можно не снимать"
```

Более полезным использованием метода является вывод имен объекта и метода, а также текста с полезной информацией о состоянии до и после выполнения интересующего оператора. Вот метод, который определен в условном листе Sheet1 и использует вызов Debug.Print:

```
Public Function CalculateFuelConsumed(ByVal Duration As Double, _
  ByVal GallonsPerHour As Double) As Double

  Debug.Print "Вошли в Sheet1.CalculateFuelConsumped"

   CalculateFuelConsumed = Duration * GallonsPerHour

  Debug.Print "Выходим из Sheet1.CalculateFuelConsumped, результат = " & _
    CalculateFuelConsumed & " галлонов"

End Function
```

Предыдущий метод вычисляет объем топлива, которое потребляется за время работы продукта, а также объем топлива, потребляемого за один час. Вызов Debug.Print используется для создания вывода в окне Immediate (Проверка), показанного на рис. 7.1.

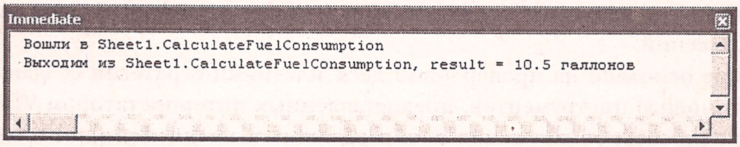


Рис. 7.1. Результат работы метода Debug.Print в окне Проверка

Помните, что объект Debug работает только в редакторе VBE. В результате отладочный код можно не удалять, так как пользователи все равно не узнают о его существовании. Преимуществом такого подхода является наличие доступности кода при возникновении любой проблемы. При этом разработчики избавляются от необходимости добавлять и удалять код между циклами отладки и сопровождения.

Метод Debug.Print еще будет рассмотрен в разделе “Отслеживание ошибок”.

Использование метода Debug.Assert

Еще одним методом объекта Debug является метод Assert. Этот метод выступает в роли наемного стрелка на стороне программиста и предназначен для остановки выполнения кода в среде разработки, если поставленное методом Assert условие не выполняется.

При создании кода программисты строят явные и неявные предположения о состоянии приложения на определенных этапах. Примером явного предположения является проверка существования файла перед его открытием для чтения. Неявным предположением является неповрежденность дискового кластера, в котором хранится необходимый файл. В любом случае, программист знает о явных и неявных предположениях только при создании кода и появлении “наглой морды” неявного предположения. Если предполагается, что что-то плохое может произойти, лучше сразу нанять стрелка, который будет патрулировать окрестности. Метод Debug.Assert является лучшим стрелком, нанятым программистом на Excel VBA.

При создании каждого метода желательно добавить условный код, проверяющий соответствие некоторых независимых параметров преполагаемым условиям. Например, деление на 0 возникать не должно. Поэтому при выполнении деления желательно убедиться, что делитель не равен 0. Но так как деление на 0 никогда не должно происходить, желательно проинструктировать стрелка о существовании этого предположения. Помните, что объект Debug является инструментом программиста. Метод Debug.Assert завершит работу приложения, если условие не будет выполняться, но объект создается только при запуске кода в редакторе VBE в процессе отладки. По этой причине объект Debug никогда не станет полноценной заменой нормальным проверкам. Он просто, с точки зрения программиста, дополняет проверки. В отличие от вызова метода Debug.Assert условие If не остановит работу приложения, если условие не выполняется. Следовательно, условный оператор If может защитить код от взрыва, а объект Debug сообщает о возможности взрыва во время отладки. В комбинации эти инструменты позволят получить более мощные средства отладки, как показано в следующем фрагменте кода:

```
Public Sub DivideByZero(ByVal Numerator As Double, _
  ByVal Denominator As Double)

  Dim result As Double

  Debug.Assert Denominator <> 0
  If (Denominator <> 0) Then
    result = Numerator / Denominator
  Else
    ' сделать что-то другое
  End If
End Sub
```

Если предположение оказывается неверным в редакторе VBE, то выполнение остановится на строке Debug.Assert Denominator <> 0. Программист сразу поймет, что потребитель (программист, который воспользовался методом DivideByZero) нарушил необходимое условие и присвоил параметру Denominator значение 0 (рис. 7.2).

В следующем разделе будет показано, как использовать методы Debug.Print и Debug.Assert для создания инструментария многоразового использования, чтобы получить защищенный код. Если время на чтение ограничено, можно пропустить разделы “Краткая история отладки на ПК” и сразу переходить к разделу “Создание многоразовых инструментов на основе объекта Debug”.

Краткая история отладки на ПК

История очень важна, так как позволяет охватить проблему в более широкой перспективе. Интересно, но история микрокомпьютеров насчитывает не более 20 лет. (Первый компьютер IBM PC поступил в продажу в августе 1981 года.) Таким образом, работавшие в 1981 году программисты являются живыми свидетелями сжатой истории микрокомпьютеров.

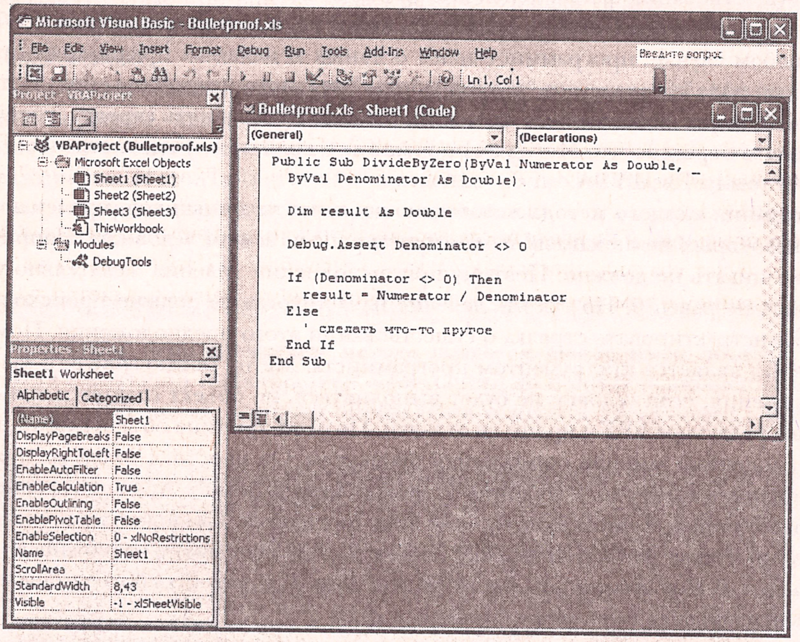


Рис. 7.2. Остановка приложения при неподтвердившемся предположении

Примерно в то время Тим Патерсон (Tim Paterson) продал собственную дисковую операционную систему Биллу Гейтсу за $25000. Впоследствии она стала называться MS-DOS. Персональный компьютер работает под управлением комбинации из дисковой операционной системы и программы BIOS (система базового ввода-вывода). Эти два компонента предоставляют базовые строительные блоки, позволяющие компьютеру работать, а программистам — создавать программы.

Программа BIOS обеспечивает работу прерываний, которые представляют собой глобальные функции, загружаемые вместе с базовой подсистемой ввода-вывода во время включения компьютера. Эти функции существуют до сих пор под покровом нескольких слоев сложного кода Windows. Прерывания имеют номера 0, 1, 2, 3 и т.д. Можно предположить, что прерывания с меньшими номерами появились раньше, а прерывания с большими номерами появились позже, вместе с появлением дополнительных возможностей. Например, прерывание DOS имеет номер 0x21 (обычно для нумерации прерываний используются шестнадцатеричные числа; данное прерывание имеет десятичный номер 33). Учитывая предположение о первоочередном появлении прерываний с меньшими номерами, не удивительно, что прерывание с номером 0 соответствует ошибке де-

ления на ноль. Деление является важной операцией для программистов, а деление на 0, похоже, было значительной проблемой, если оно обрабатывается одной из самых базовых системных служб.

Назначение прерываний 1 и 2 мало кого интересует. Прерывание 3 оказывается более важным с точки зрения текущего обсуждения, так как это прерывание отладки. Оно останавливает выполнение программы. Скорее всего, именно это прерывание лежит в основе оператора `Stop` в языке VBA и точек останова в редакторе VBE. (В более ранних версиях отладочных инструментов прерывание 3 называлось “soft-ice”; то есть, вызов этого прерывания приводил к “замораживанию” программы.)

Все эти базовые возможности до сих пор существуют и используются компьютером для выполнения поставленных ранее задач, хотя над этими прерываниями и создан большой объем кода. Убедиться в существовании этих возможностей можно с помощью команды `debug.exe` в приглашении командной строки. Для этого можно написать простой машинный код. (Будьте осторожны, так как базовые возможности обладают большой мощностью.)

Руководствуясь рис. 7.3, введите следующие отладочные инструкции и код ассемблера. В результате будет создана программа на языке ассемблера. (Этот пример является хорошим напоминанием о том, насколько более удобным инструментом является язык VBA.)

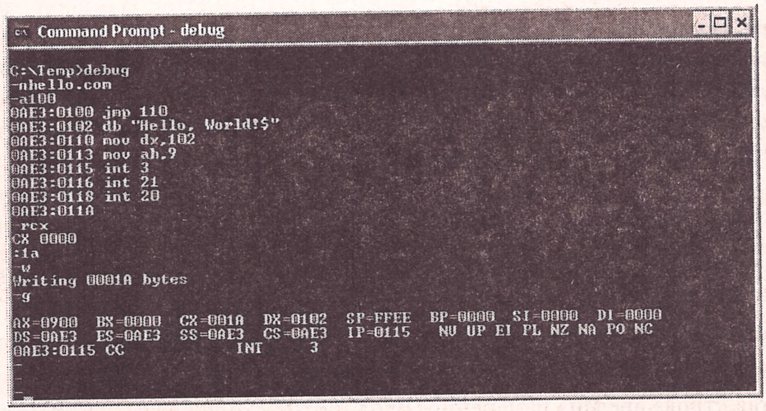


Рис. 7.3. Ввод программы в отладчике debug.exe

```
debug
nhello.com
a100
jmp 110
db "Hello, World!$"
mov dx,102
mov ah,9
int 3
int 21
int 20

rcx
1a
w
g
q
```

Ниже по порядку приводятся инструкции и код.

- Запустите программу Debug. Для этого в приглашении командной строки введите команду **debug.exe**. Это очень простой отладчик и редактор, который иногда может оказаться очень мощным.
- Введите **nhello.com**. Эта инструкция сообщает редактору debug.exe имя файла, в который необходимо записывать вывод.
- Введите **a100**. Это инструкция языка ассемблера. Она сообщает редактору, что начинается ввод кода.
- Введите **jmp 110**. Эта команда является аналогом нашего хорошего друга — команды GOTO на языке ассемблера.
- Команда **db "Hello, Word!$"** используется для объявления строковой переменной.
- Команда **mov dx,102** заносит значение 102 (шестнадцатеричное значение) в регистр DX центрального процессора. Регистры являются аналогами переменных на самом нижнем уровне. Вообще это переключатели внутри микропроцессора, которые используются миллион раз в секунду. На этом этапе в регистр DX заносится адрес текстовой строки.
- Команда **mov ah,9** в данном контексте является функцией. В целом эти команды используются для подготовки вызова функции, аргументом которой является строка "Hello, Word!" (символ $ является символом окончания строки), а 9 является номером функции.
- Команда **int 3** выполяет функцию точки останова. Код будет выполняться до команды **int 3**, после чего выполнение будет приостановлено, и выдано текущее состояние процессора. (Текущее состояние показано на рис. 7.3.)
- Команда **int 21** является прерыванием DOS. Прерывание 21 и функция 9 используются для вывода строк.
- Вызов прерывания **int 20** приводит к завершению работы программы. Он сообщает процессору о необходимости передать управление операционной системе.
- Пустая строка введена намеренно. Ввод пустой строки возвращает редактор из режима программирования в режим управления.
- Команда **rcx** является инструкцией отладчика, позволяющей вывести и отредактировать значение регистра CX, который используется, чтобы сообщить отладчику количество записываемых байт.
- **1a** является шестнадцатеричным числом (в десятичной форме 26). Отладчик должен записать 26 байт ассемблерного кода.
- Команда **w** является инструкцией на запись.
- Команда **g** является аналогом клавиши <F5> в среде разработки VBA. Эта команда приводит к выполнению программы в режиме отладчика.
- Команда **q** приводит к выходу из программы. Введите **q** и нажмите клавишу <Enter>. Работа утилиты debug.exe завершится.

После завершения работы утилиты будет создана программа `hello.com`. Файл будет храниться во временном каталоге или в каталоге, который был указан в начале работы программы. Введите **`Hello`** в приглашении командной строки. Обратите внимание на выведенный текст `Hello World!`. После этого программа возвращает управление интерпретатору командной строки. Кроме этого, обратите внимание на то, что точка останова была проигнорирована. Не кажется ли знакомым это поведение? Да, верно. Оно характерно для объекта `Debug`, который создается только при отладке в редакторе VBE. За пределами редактора VBE вызовы методов объекта `Debug` игнорируются. Можно считать, что принципы отладки были показаны до самого нижнего, аппаратного, уровня. (Ниже только физические механизмы в кристалле процессора, но рассматривать их будет лишним.) Теперь понятно, почему деление на ноль и точки останова были так важны для ранних программистов. Кроме этого, ясно, что эти возможности все еще используются в компьютере, хотя они и скрыты за более удобными средами вроде VBE.

Создание многоразовых инструментов на основе объекта Debug

Одной из любимых авторами книг по программированию является *No Bugs!* Дейва Тилена (Dave Thielen), которая выпущена издательством Addison-Wesley. Книга для программистов на C была написана после успешного выхода операционной системы MS-DOS 5.0. Хотя C является языком более низкого уровня по сравнению с VBA, большинство приемов отладки основаны на тех же принципах, поэтому данные приемы можно использовать в VBA вместе с объектом `Debug`. Важность этих подходов связана с тем, что они работают всегда и их работоспособность проверена временем.

В этом разделе рассматривается три мощных управляемых инструмента, которые позволяют систематически исправлять ошибки и быть уверенными, что обнаруженные ошибки исправлены. Если что-то пойдет не так, эти инструменты обнаружат и исправят проблему немедленно. Они называются `Trace`, `Trap` и `Assert`. Как компания Microsoft создала объект `Debug` и оператор `Stop` поверх базовых возможностей BIOS (как показано в разделе "Краткая история отладки на ПК"), так и мы создадим нашу реализацию поверх объекта `Debug` и оператора `Stop`.

Определение последовательности выполнения

Определение последовательности обеспечивается размещением строк кода, сообщающих информацию о маршруте и состоянии кода в определенный момент. Эта информация может оказаться настолько полезной в процессе отладки, что многие инструменты разработчика автоматически отслеживают выполнение кода, создавая стек вызовов (определяя последовательность вызова методов) и показывая последовательность выполнения. Для определения последовательности выполнения в VBA может использоваться метод `Debug.Print`, но автоматическое отслеживание порядка вызова методов при этом не поддерживается. Это придется делать вручную. Можно создать собственную версию подпрограммы `Trace` и описать интересующую информацию, получая одинаковый по форме вывод при каждом использовании этого инструмента. Вот метод `Trace`, реализованный в экспортируемом модуле `DebugTools.vb`.

```
Option Explicit
#Const Tracing = True
#Const UseEventLog = False

Private Sub DebugPrint(ByVal Source As String, _
  ByVal Message As String)

#If UseEventLog Then

#Else
  Debug.Print "Источник: " & Source & " Сообщение: " & Message
#End If

End Sub
Public Sub Trace(ByVal Source As String, _
  ByVal Message As String)
#If Tracing Then
  Call DebugPrint(Source, Message)
#End If

End Sub
```

В предыдущем фрагменте кода была определена константа компилятора: `Tracing`. Метод `DebugPrint` является универсальным методом, который принимает в качестве параметров две строки. Параметры называются `Source` и `Message`. Обратите внимание на закомментированную часть кода, в которой константа `UserEventLog` управляет вызовом метода `Debug.Print`. (К условию, зависящему от константы `UserEventLog`, возвратимся в разделе "Запись в журнал событий" далее в этой главе.) Последний метод, `Trace`, принимает те же два аргумента: `Source` и `Message`. Метод `Trace` проверяет значение константы отладчика. Если константа равна `True`, вызывается процедура `DebugPrint`.

Можно спросить: зачем использовать константы компилятора и заворачивать вызов метода `Debug.Print`, если объект `Debug` автоматически отключается при запуске кода за пределами VBE? Хороший вопрос.

Программирование очень похоже на выращивание лука изнутри. Программирование начинается с ядра, например базовой подсистемы ввода-вывода компьютера. Постепенно слои накладываются поверх существующих уровней, медленно, но уверенно добавляя сложности. (Сложность может оказаться субъективным параметром.) Эти уровни должны быть небольшими, простыми для тестирования и предоставлять необходимый уровень функциональности, которая не должна быть сложной или большой по объему. Причиной постепенного добавления функциональности является попытка снизить сложность реализации изменений и минимизация вероятности появления дополнительных дефектов. Кроме этого, простота позволяет легко оценить полезность вносимых изменений. Другими словами, большие объемы кода (большие монолитные реализации) являются причиной головных болей в будущем.

Однозначное поведение, построенное поверх более простого однозначного поведения, как ламинированное дерево, позволяет построить мощные и полезные реализации, которые сохраняют простоту на каждом уровне. Именно так выглядит хороший, полезный, поддающийся отладке и защищенный код. К сожалению, эта особенность программирования не рассматривается в институтах и университетах. (Просим извинения у тех, чье учебное заведение было исключением.) Это связано с тем, что данная отрасль все еще молода

и не все еще приняли правильное решение по этому вопросу. (Стоит обратить внимание, что большинство успешных специалистов в данной области все-таки используют ту или иную аналогию со слоями лука.) Создание многоуровневого, однозначного кода требует определенной предварительной практики, но именно для этого написана данная книга.

Помня о необходимости создания сложных систем в виде нескольких простых уровней следует создать простой метод Trace. Метод принимает два строковых параметра. Кроме этого, метод предоставляет отдельную возможность включения и отключения отладки. На основе поведения редактора VBE определение отслеживания вызовов было расширено до использования другого хранилища. Вместо ведения журнала в окне Immediate (Проверка) был добавлен код для записи журнала в более постоянное хранилище, в журнал событий. После этого необходимо научиться использовать новое поведение метода Trace.

Программист должен выбрать отслеживаемые параметры приложения. Можно отслеживать все параметры, но это слишком утомительно. Вместо этого необходимо добавить вызовы Trace в тех местах, которые особенно важны для реализации, а также в местах, где возникают ошибки. После добавления вызова Trace его можно оставить даже после исправления всех ошибок. При внесении изменений в код могут появиться новые ошибки, но отладочный код уже будет присутствовать в ключевых местах. Одним из преимуществ вызова Trace является обозначение фрагментов кода, которые интересовали разработчиков в прошлом. То есть, вызов Trace выступает в качестве "хлебных крошек", которые разработчики оставляют для того, чтобы вернуться обратно, если возникнет необходимость. Вот метод CalculateFuelConsumed из предыдущей главы. Вызовы метода DebugPrint заменены на вызовы Trace:

```
Public Function CalculateFuelConsumed(ByVal Duration As Double, _
  ByVal GallonsPerHour As Double) As Double

  Call Trace("Sheet1.CalculateFuelConsumped", _
    "Длительность=" & Duration & " Галлонов в час =" &
↳ GallonsPerHour)

  CalculateFuelConsumed = Duration * GallonsPerHour
  Call Trace("Sheet1.CalculateFuelConsumped", "Результат=" &
↳ CalculateFuelConsumed)

End Function
```

Конечным результатом являются одинаковые вызовы Trace и одинаковые правильно отформатированные сообщения. На рис. 7.4 показано окно Immediate (Проверка) после вызова метода CalculateFuelConsumed.

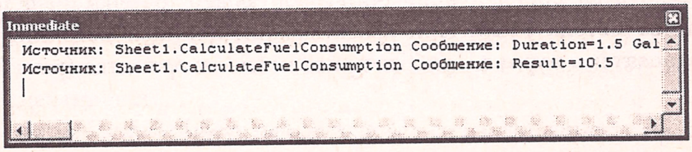


Рис. 7.4. Результат использования метода Trace

Получение маршрута выполнения кода

Следующий метод называется перехватом. Необходимо проверять все маршруты, по которым может выполняться код — условия If и Else, все методы и свойства, циклы While, тело которых выполняется и не выполняется. Если этого не сделать, то нельзя быть уверенным, что непроверенный маршрут не приведет к опасному поведению приложения.

Идиома Trap используется для отметки веток кода, чтобы обеспечить тестом каждый закоулок кода и получить ожидаемый и необходимый результат. Идиома Trap находится уровнем выше оператора Stop и, как и метод Trace, помещается в оболочку для получения дополнительной гибкости. Вот код реализации идиомы Trap, который можно добавить в модуль DebugTools. После этого будет показано, как создавать ловушки:

```
Option Explicit
#Const Tracing = True
#Const Debugging = True
#Const UseEventLog = False

Private Sub DebugPrint(ByVal Source As String, _
  ByVal Message As String)

#If UseEventLog Then

#Else
  Debug.Print "Источник: " & Source & " Сообщение: " & Message
#End If

End Sub
Public Sub Trap(ByVal Test As Boolean, _
  ByVal Source As String, _
  ByVal Message As String)

#If Debugging Then
  If (Test) Then
      Call DebugPrint(Source, Message)
      Stop
    End If

#End If
End Sub
```

В подпрограмме Trap используется константа Debugging и метод DebugPrint, уже применяемый ранее. Не будем еще раз описывать назначение этих элементов. Рассмотрим особенности метода Trap.

В методе Trap используются те же два аргумента, что и в методе Trace. Это Source и Message. Они нужны для идентификации сработавшей ловушки и для поиска соответствующего вызова метода Trap. При срабатывании ловушки источник срабатывания и сообщения заносятся в журнал, а работа приложения завершается (рис. 7.5).

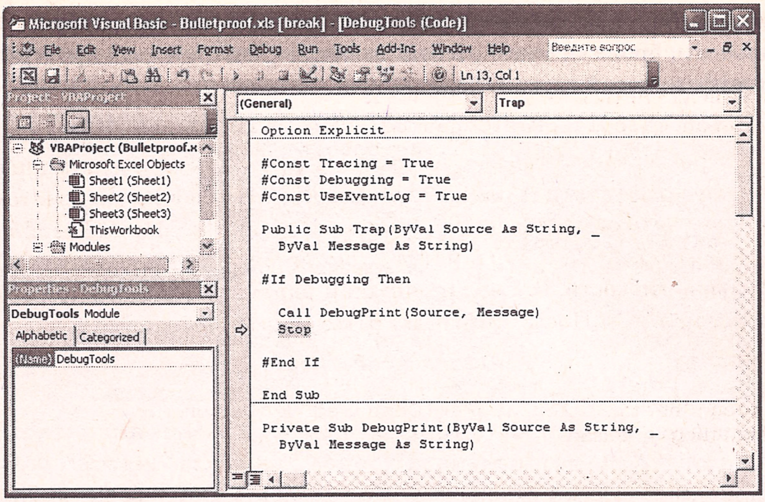


Рис. 7.5. Остановка приложения при срабатывании ловушки

После срабатывания ловушки значение параметра `Source` можно использовать для поиска и комментирования сработавшего вызова. Комментирование конкретного вызова метода `Trap` указывает, что данный маршрут выполнения кода был протестирован и можно переходить к тестированию других веток кода. При этом закомментированный вызов метода `Trap` остается внутри кода на случай, если тестирование придется повторить. Например, для проверки существования нового теста для метода `CalculateFuelConsumed` в этот метод можно добавить вызов метода `Trap`. Вот как будет выглядеть модифицированная версия метода `CalculateFuelConsumed`:

```
Public Function CalculateFuelConsumed(ByVal Duration As Double, _
  ByVal GallonsPerHour As Double) As Double

  Call Trap("Sheet1.CalculateFuelConsumed", "Tested")

Call Trace("Sheet1.CalculateFuelConsumed", _
    "Длительность=" & Duration & " Галлонов в час =" &
↳ GallonsPerHour)

  CalculateFuelConsumed = Duration * GallonsPerHour

  Call Trace("Sheet1.CalculateFuelConsumed", "Результат=" &
↳ CalculateFuelConsumed)

End Function
```

После создания теста для метода `CalculateFuelConsumed` оператор вызова метода `Trap` можно закомментировать.

Чистоплотные разработчики могут не согласиться со смешиванием отладочного кода и кода, выполняющего поставленную перед приложением задачу. В таком случае лучше принять стратегию разделения между реальным и отладочным кодом. Здесь будет использоваться стратегия создания закрытого теневого метода с префиксом `Do`. Реальный код добавляется в метод с префиксом `Do`, а в методе с обычным именем будет находиться

отладочный код. В результате выполнение алгоритма будет происходить отдельно от функциональности бронежилета — отслеживания, перехвата и кода проверки предположений. Вот исправленный вариант кода:

```
Public Function CalculateFuelConsumed(ByVal Duration As Double, _
  ByVal GallonsPerHour As Double) As Double

  'Call Trap("Sheet1.CalculateFuelConsumed", "Tested")

  Call Trace("Sheet1.CalculateFuelConsumed", _
    "Длительность=" & Duration & " Галлонов в час =" &
↳ GallonsPerHour)

  CalculateFuelConsumed = DoCalculateFuelConsumed(Duration, _
                          FuelConsumedPerhour)

  Call Trace("Sheet1.CalculateFuelConsumed", "Результат =" &
↳ CalculateFuelConsumed)

End Function

Private Function DoCalculateFuelConsumed(ByVal Duration As Double, _
  ByVal GallonsPerHour As Double) As Double

  DoCalculateFuelConsumed = Duration * GallonsPerHour

End Function
```

Неискушенный наблюдатель может решить, что дополнительный код выглядит неуклюже. Так и есть. Но тут, как и в случае с бронежилетом, стоит обеспечить максимальную дополнительную защиту. В итоге подобная структура кода продемонстрирует профессиональный подход к решению задачи. После завершения создания отладочного инструментария DebugTools можно рассчитывать на формирование привычки. А сформированная привычка значительно упростит написание такого кода. Если вам повезло работать с программистами различного уровня, то вы можете создавать реализацию фактического поведения, а младший программист может заворачивать его в оболочку отладочного и тестового кода. Кроме этого, средства отладки могут быть интегрированы в код при появлении проблем.

Проверка инвариантных условий

Наконец пришло время рассмотреть метод проверки предположений. Так получилось, что нам нравятся полицейские, но если кого-то беспокоит аналогия с органами охраны правопорядка, помните, что программирование можно представить как инструктирование компьютера и определение границ контроля над выполнением. Без открытых и закрытых методов не было бы методов защиты данных. Без приемов отладки не было бы способов решения проблем. Поэтому программист, как полицейский на параде, старается предотвращать проблемы и следить за правильной работой кода.

Предположение является обученным, одетым в бронежилет и форму, полицейским офицером. При создании кода программист делает базовые предположения о состоянии и поведении кода. Поведение Assert позволяет обеспечить сохранение истинности предположений. Результат похож на утопию, где нет вождения в нетрезвом виде, нецензурной брани и хулиганских выходок.

Метод Assert стоит использовать в тех местах кода, где предположения всегда должны быть истинными. Для сохранения однородности заключите метод Assert в оболочку DebugTools. Это позволит определиться с ожиданиями и обнаружить места, где ожидания не оправдываются. Вот пример кода:

```
Public Sub Assert(ByVal Test As Boolean, _
  ByVal Source As String, ByVal Message As String)

#If Debugging Then
  If (Not Test) Then
      Call DebugPrint(Source, Message)
      Debug.Assert False
    End If
#End If

End Sub
```

Метод Assert использует константу Debugging, которая определена ранее в этой главе, и метод DebugPrint. Метод Assert принимает параметр типа Boolean и еще два параметра: Source и Message. Сначала необходимо убедиться, что приложение работает в режиме отладки. После этого проверить истинность или ложность предположений. Если предположение не оправдалось, в журнал записывается сообщение и вызывается базовый метод Assert, который приводит к остановке работы приложения. Теперь рассмотрим проверку предположений в методе CalculateFuelConsumed. В данном случае необходимо убедиться, что параметры GallonsPerHour и Duration имеют неотрицательные значения. (В метод Do также добавлены условные операторы.)

```
Public Function CalculateFuelConsumed(ByVal Duration As Double, _
  ByVal GallonsPerHour As Double) As Double

  'Call Trap("Sheet1.CalculateFuelConsumption", "Tested")

  Call Trace("Sheet1.CalculateFuelConsumed", _
    "Длительность =" & Duration & " Галлонов в час =" & _
 GallonsPerHour)

  Call Assert(Duration > 0, "Sheet1.CalculateFuelConsumed", _
"Duration > 0")
  Call Assert(GallonsPerHour > 0, "Sheet1.CalculateFuelConsumed", _
    "GallonsPerHour > 0")

  CalculateFuelConsumed = DoCalculateFuelConsumed(Duration, _
FuelConsumedPerhour)

  Call Trace("Sheet1.CalculateFuelConsumption", "Результат =" &
↳ CalculateFuelConsumed)

End Function
Private Function DoCalculateFuelConsumed(ByVal Duration As Double, _
  ByVal GallonsPerHour As Double) As Double

  If (Duration > 0 And GallonsPerHour > 0) Then
    DoCalculateFuelConsumed = Duration * GallonsPerHour
  Else
    ' Здесь выдается сообщение об ошибке
  End If

End Function
```

В модифицированном примере добавлены два вызова метода Assert. Первый вызов сравнивает значение параметра Duration с 0. Обратите внимание, что в методе DoCalculateFuelConsumed добавлена проверка времени выполнения. В результате метод стал самодокументированным, имя описывает назначение, а известные механизмы тестирования делают метод достаточно защищенным. Вызов метода Trap добавлен для напоминания о необходимости тестирования. Глобальный поиск метода Trap позволяет найти незакомментированные вызовы и обнаружить непротестированные методы. Здесь используется два оператора Trace, которые показывают, где и в каком порядке используется этот метод. Наконец, добавлены предположения о допустимости значений параметров.

Вот еще один совет перед тем, как переходить к следующей теме. Что делать, если после написания определенного фрагмента кода и добавления вызовов Trap и Trace оказывается, что ни одна из ловушек не срабатывает? Ответ предполагает сброс мертвого балласта. Другими словами, удалите ненужный код. Он просто занимает дорогое время разработчиков, которые вынуждены читать на самом деле не представляющий интереса код.

Обратите внимание, что последний вариант метода содержит два оператора ветвления и блок Else, содержащий только комментарий. Необходимо добавить вызов Trap в каждую ветку условного оператора, чтобы протестировать оба маршрута выполнения кода. Кроме этого, нужно рассмотреть создание сообщения об ошибке. (Сообщения об ошибках будут рассматриваться в следующем разделе.) Далее показана последняя версия кода в этом разделе, в которой продемонстрирован удобный способ добавления операторов Trap в каждую ветку условного оператора:

```
Public Function CalculateFuelConsumed(ByVal Duration As Double, _
  ByVal GallonsPerHour As Double) As Double

  'Call Trap("Sheet1.CalculateFuelConsumption", "Tested")

  Call Trace("Sheet1.CalculateFuelConsumed", _
    "Длительность =" & Duration & " Галлонов в час =" &
➥ GallonsPerHour)

  Call Assert(Duration > 0, "Sheet1.CalculateFuelConsumed", _
"Duration > 0")
  Call Assert(GallonsPerHour > 0, "Sheet1.CalculateFuelConsumed", _
    "GallonsPerHour > 0")

  If (Duration > 0 And GallonsPerHour > 0) Then
    Call Trap("Sheet1.CalculateFuelConsumed", _
      "Duration > 0 And GallonsPerHour > 0")
    CalculateFuelConsumed = DoCalculateFuelConsumed(Duration, _
GallonsPerHour)
  Else
    Call Trap("Sheet1.CalculateFuelConsumed", "Duration > 0 And
➥ GallonsPerHour > 0 is False")
    ' Здесь выдается сообщение об ошибке
  End If

    Call Trace("Sheet1.CalculateFuelConsumption", "Result=" &
➥ CalculateFuelConsumed)

End Function

Private Function DoCalculateFuelConsumed(ByVal Duration As Double, _
  ByVal GallonsPerHour As Double) As Double
```

```
DoCalculateFuelConsumed = Duration * GallonsPerHour

End Function
```

Обратите внимание, что в последнем варианте кода вместе с предположениями используются операторы `If`, а весь отладочный код расположен во внешнем методе (не имеющем префикса `Do`). В результате алгоритм оказывается очень ясным и полностью защищенным.

Кто-то может поинтересоваться, нужно ли писать этот код для каждого метода? Ответ: нет. После написания пары миллионов строк кода появилась уверенность в том, что однострочный метод настолько прост в тестировании и отладке, что добавлять такой объем отладочного кода к столь простому методу абсолютно излишне. На самом деле, стоит пытаться сохранить простоту кода на уровне метода `DoCalculateFuelConsumed`. Кроме избежания большого количества ошибок такая простота избавляет от необходимости писать комментарии или отладочный код. Начинающие разработчики еще не так уверены в своих силах, а попав в трудную ситуацию, начинают сомневаться в правильном выборе путей выхода из нее. В итоге сам разработчик принимает решение о достаточном объеме отладочного кода. Это субъективное решение, которое и делает хорошее программирование таким сложным.

Вывод сообщений об ошибках

Существует много хороших программистов, которые не согласны с нашими рекомендациями. Это хорошо. Именно это и делает жизнь интересной и позволяет создавать новые идеи. Есть одна область, в которой сотни и даже тысячи программистов не могут прийти к единому решению. Надеемся убедить читателей в нашей правоте и в ошибочности подхода других программистов.

Много лет назад семантически более слабые языки, например С, возвращали коды ошибок при любом некорректном поведении. Коды ошибок представляли собой произвольные целые числа (обычно отрицательные), имевшие смысл только в определенном контексте. В итоге был создан большой объем кода, возвращавшего определенное целое число, имевшее смысл в определенном контексте. Такой подход к программированию предполагал, что все является функцией, все реальные возвращаемые значения передавались по ссылке (`ByRef`), а смысл кода ошибки не передавался вместе с ошибкой. Знание о смысле кода хранилось отдельно. Таким образом, если функция возвращала значение –1 в качестве кода ошибки, потребитель кода должен был обращаться к другому источнику для получения смысла кода ошибки –1. Вот переработанная версия метода `DoCalculateFuelConsumed`, которая возвращает код ошибки. (Такой код писать не рекомендуется.)

```
Private Function CalculateFuelConsumed(ByVal Duration As Double, _
  ByVal GallonsPerHour As Double, ByRef Result As Double) As Integer

  If (Duration > 0 And GallonsPerHour > 0)Then
    Result = Duration * GallonsPerHour
    CalculateFuelConsumed = 0
  Else
    CalculateFuelConsumed = -1
  End If

End Function
```

В данной версии метода возвращаемое значение указывает на успешность завершения работы, а фактический результат работы алгоритма передается через параметр. Это усложняет непосредственное использование функции, так как необходимо объявить дополнительный аргумент для хранения и получения результата работы. Теперь для вызова метода CalculateFuelConsumed необходимо писать значительно больше кода.

```
Dim Result As Double
If( CalculateFuelConsumed(7, 1.5, Result) = 0) Then
' Да: все сработало и Result можно использовать
Else
  ' Жаль, опять неудача и содержимое Result не имеет смысла
End If
```

Это особенно пессимистический подход к программированию. Полезность возвращаемого значения функции практически исчезает и приходится программировать так, как будто код всегда находится на грани падения. Продолжив чтение главы о создании защищенного кода, вы поймете, что код будет отказывать значительно реже. Вернемся к использованию возвращаемого значения функции и создадим оптимистический код, предполагая, что отказ является редкой неприятностью, а не частым гостем. Для этого можно отказаться от применения кодов ошибок и перейти к выдаче сообщений об ошибках только тогда, когда они действительно возникают.

```
Option Explicit

Public Function CalculateFuelConsumed(ByVal Duration As Double, _
  ByVal GallonsPerHour As Double) As Double

  If (Duration > 0 And GallonsPerHour > 0) Then
    CalculateFuelConsumed = Duration * GallonsPerHour
  Else
    Call CalculateFuelConsumedError(Duration, GallonsPerHour)
  End If

End Function

Private Sub CalculateFuelConsumedError(ByVal Duration As Double, _
  ByVal GallonsPerHour As Double)

  Const Source As String = "Sheet2.CalculateFuelConsumed"
  Dim Description As String

  Description = "Длительность {"
➥& Duration & "} и Галлонов в час {"
➥& GallonsPerHour & "} должны быть больше 0"

  Call Err.Raise(vbObjectError + 1, Source, Description)
End Sub
Public Sub Test()

  On Error GoTo Catch
    MsgBox CalculateFuelConsumed(-8, 1.5)

  Exit Sub
Catch:
  MsgBox Err.Description, vbCritical

End Sub
```

Функция `CalculateFuelConsumed` выполняет все рассчеты. Если параметры `Duration` и `GallonsPerHour` имеют некорректное значение, сообщение об ошибке выдается через вызов вспомогательной функции `CalculateFuelConsumedError`. Впомогательный метод создает удобно отформатированное описание и выдает сообщение об ошибке при помощи вызова метода `Err.Raise` и передачи необязательного кода ошибки, исходной строки, описания, имени файла справочного руководства и контекста файла справочного руководства в качестве параметров. По соглашению контекстные номера ошибок добавляются в константу `vbObjectError` (как показано ранее). Это позволяет предотвратить пересечение собственных номеров ошибок и номеров ошибок языка VBA.

Для использования метода `CalculateFuelConsumed` создается оператор перехода на метку `On Error GoTo`. Стоит использовать такой способ (`Catch` или `Handle`) постоянно. После этого добавляется вызов метода и оператор выхода из метода: `Exit Sub` для подпрограмм, `Exit Function` для функций или `Exit Properties` для свойств. Наконец, в конце с оптимизмом добавляется метка и код для обработки ошибки. В этом примере пользователь получает сообщение об ошибке, выводимое с помощью оператора`MsgBox`.

Если кого-то интересует, почему этот подход лучше других, можно отметить, что:

- функцию можно использовать, как и предполагалось, для возврата подсчитанного результата;
- не добавляется код ошибки “ничего не делать”, который пессимистически использует условный оператор для проверки правильности завершения функции. Можно предполагать, что все работает как надо;
- присутствует страховочная сетка, которая перехватывает любые ошибки, а не только собственные коды ошибок. (Например, можно не проверять неравенство делителя нулю. Если будет выполнено деление на ноль, при этом используется встроенный код ошибки. При пессимистическом подходе эта ошибка прошла бы сквозь систему контроля. Оператор `On Error GoTo` такие ошибки не пропускает);
- объект `Error` передает значение ошибки вместе с сообщением об ошибке. Разработчик избавляется от необходимости создавать смысл ошибки на основе произвольного значения кода ошибки.

Если один стиль лучше с одной стороны, то он лучше и в общем. Выдача сообщений об ошибках лучше возврата кодов ошибок по нескольким несубъективным критериям. Теперь стоит рассмотреть создание обработчиков ошибок.

Создание обработчиков ошибок

Существует три формы оператора `On Error`. Это оператор перехода на произвольную строку кода `On Error GoTo`. Далее оператор `On Error Resume Next`, обеспечивающий переход на строку сразу после строки, которая привела к появлению ошибки. Кроме этого, еще существует оператор `On Error GoTo 0`, просто сбрасывающий сообщение об ошибке.

Оператор On Error GoTo

Оператор `On Error GoTo` уже использовался вместе с меткой `Catch` в предыдущем разделе. Важно, чтобы метод завершал нормальную работу до метки. Иначе метка и код обработки ошибки будут выполняться даже при нормальной работе. Для завершения ра-

боты подпрограммы применяется оператор Exit Sub. Для завершения работы функции — оператор Exit Function, а для завершения работы свойства — оператор Exit Property. Но что, если код обработки ошибок должен работать всегда? Такое бывает.

Подобный прием называется блоком защиты ресурсов. Компьютеры используют конечное количество сущностей, которые в целом называются ресурсами. Ресурсом может быть подключение к базе данных, файлу или сетевому сокету. Если создаются экземпляры таких ресурсов, то необходимо обеспечить их правильное удаление. Для этого можно воспользоваться идиомой блока защиты ресурсов. Она имеет простой принцип работы: использовать оператор On Error GoTo сразу после создания ресурса и намеренно не добавлять оператор завершения процедуры перед соответствующей меткой. Таким образом, код обработки ошибок (в данном случае это код защиты ресурсов) будет выполняться вне зависимости от наличия или отсутствия ошибок. Вот как выглядит реализация идиомы для открытия файла и записи текста в открытый файл. (Не стоит использовать такой способ записи в файлы. Для этого имеет смысл применять объект FileSystemObject.)

```
Public Sub ProtectThisResource()

  Open ThisWorkbook.Path & "\dummy.txt" For Output As #1
  On Error GoTo Finally

    Print #1, "Этот файл всегда будет закрыт"

Finally:

  Close #1

  If (Err.Number <> 0) Then MsgBox Err.Description

End Sub
```

> Помните, что в качестве меток могут использоваться произвольные номера строк. Для этого в операторе On Error GoTo строку метки можно заменить на число.

Базовая последовательность действий блока защиты ресурсов состоит из создания ресурса, добавления оператора On Error GoTo, попытки использования ресурса и очистки (ресурса или ошибки). В этом примере открывается текстовый файл, выполняется оператор On Error GoTo Finally, делается попытка использования ресурса и, в любом случае, выполняется очистка. Обратите внимание на отсутствие оператора выхода из процедуры.

Оператор On Error Resume Next

Оператор On Error Resume Next может использоваться для игнорирования ошибки и продолжения выполнения со следующего оператора. Этот способ применяется непосредственно перед оператором, который может привести к появлению ошибки (например, перед оператором, результат выполнения которого не особенно важен). Оператор On Error Resume Next используется редко, так как не часто создаются операторы, результат выполнения которых не важен. Если результат работы оператора никого не интересует, удалите его.

Операторы Resume и Resume Next могут использоваться сами по себе. Оператор Resume применяется в завершении процедуры обработки ошибки для повтора попытки выполнить оператор. Например, если добавить дополнительный блок обработки ошибок

в метод ProtectThisResource на случай невозможности открыть файл из-за установленного атрибута ReadOnly (Только чтение), то в процессе обработки ошибки можно сбросить этот атрибут и выполнить оператор Resume для повторения попытки открыть файл. Оператор Resume Next передает управление следующему оператору после оператора, который привел к появлению ошибки. Такой оператор используется в ситуациях, когда вызвавшую ошибку строку можно пропустить. В следующем методе показано, как блок защиты ресурсов можно использовать вместе с блоком обработки ошибок, а также как восстанавливать работоспособность после попытки открыть защищенный от записи файл:

```
Public Sub ProtectThisResource()

  Dim FileName As String
  FileName = ThisWorkbook.Path & "\dummy.txt"

    On Error GoTo Catch
  Open FileName For Output As #1

  On Error GoTo Finally
    Print #1, Time & " Этот файл всегда будет закрыт"

Finally:

  Close #1
  If (Err.Number <> 0) Then MsgBox Err.Description
  Exit Sub

Catch:
  If (Err.Number = 75) Then
    Call SetAttr(FileName, vbNormal)
    Resume
  End If

End Sub
```

Первый оператор On Error GoTo Catch передает управление обработчику защищенных от записи файлов, позволяющему открыть файл еще раз после сброса атрибутов. Второй оператор On Error GoTo обеспечивает закрытие файла после завершения работы процедуры.

Пример метода может показаться слишком сложным и стоит обратить внимание, что ошибки могут возникнуть и в других местах. Что, если поврежден диск? Что, если недостаточно памяти для загрузки слишком большого файла? Что, если файл заблокирован другим приложением? Возможны любые ошибки. Именно поэтому сложно написать стабильную программу, а также предусмотреть все возможные варианты и все условия. Программист должен субъективно оценить возможные неприятности и попытаться обойти их. Этот процесс займет некоторое время, но в итоге он все равно завершится и код будет передан пользователям. Некоторые специалисты говорят, что создание программного обеспечения компьютеров является самым сложным видом деятельности. С этим нельзя не согласиться.

> На этом этапе уже были показаны некоторые ошибки, которые могут возникать и в простом коде. Теперь представьте себе создание 10 или 20 миллионов строк защищенного кода, лежащих в основе Windows или Windows NT. На компанию Microsoft оказывается постоянное давление с целью стимулировать создание более защищенного кода Windows, но есть очень умные люди, которые наслаждаются процессом обнаружения дыр в Windows. Удивительно то, что это происходит не слишком часто.

Оператор On Error GoTo 0

Оператор On Error GoTo 0 отключает обработчики ошибок в текущей процедуре. Это еще один оператор, который используется не очень часто. Но иногда он встречается. Его можно воспринимать, как выключатель обработчиков ошибок на уровне процедуры.

Использование объекта Err

Объект Err содержит информацию о самой последней ошибке. В объекте хранится код ошибки, источник ошибки, описание ошибки и ссылка на документ справочного руководства, если таковое существует.

Объект Err является экземпляром шаблона Singleton. Это значит, что такой объект существует в единственном экземпляре в пределах приложения. Так как это экземпляр класса, он имеет свойства и методы. Для выдачи сообщения об ошибке можно воспользоваться методом Err.Raise, а для очистки сообщения — методом Err.Clear. Оставшиеся свойства (кроме одного) уже рассматривались. Они используются для инициализации ошибок. Последним нерассмотренным свойством является LastDllError. Это свойство возвращает значение типа Hresult, которое обычно возвращается из библиотек DLL. Это свойство необходимо использовать при вызове методов во внешних библиотеках DLL, например в библиотеках Windows API.

Создание обвязки

Перед тем как перейти к обсуждению журнала событий, стоит обратить внимание на то, где и когда необходимо создавать тестовый код. В данном случае используется метод обвязки (scaffolding). Если программирование похоже на добавление рассказов в общую структуру, то создание обвязки предполагает добавление текста к каждому рассказу. Создание обвязки позволяет убедиться, что новый код является изолированным и не добавляет ошибки в уже существующий проверенный код.

Например, модуль DebugTools был создан для повторного использования в других приложениях. Данный код должен быть изолирован, но в этом не будет уверенности, пока не протестирован отладочный код. Тогда необходимо добавить по одному тестовому методу для каждого открытого метода в модуле DebugTools, вызывающего код Trace, Assert и Trap. Это позволяет убедиться, что выполнение кода приводит к получению ожидаемого результата. (Было бы просто смешно, если бы отладочный код становился причиной появления ошибок.) Вот полный листинг модуля DebugTools вместе с обвязкой тестового кода, выделенной полужирным шрифтом.

```
Option Explicit

#Const Tracing = True
#Const Debugging = True
#Const UseEventLog = False

Public Sub Trap(ByVal Source As String, _
  ByVal Message As String)

#If Debugging Then

 Call DebugPrint(Source, Message)
 Stop
```

```
#End If

End Sub
Private Sub DebugPrint(ByVal Source As String, _
  ByVal Message As String)

#If UseEventLog Then

#Else
  Debug.Print "Источник: " & Source & " Сообщение: " & Message
#End If

End Sub
Public Sub Assert(ByVal Test As Boolean, _
  ByVal Source As String, ByVal Message As String)

#If Debugging Then
  If (Not Test) Then
    Call DebugPrint(Source, Message)
    Debug.Assert False
  End If
#End If

End Sub
Public Sub Trace(ByVal Source As String, _
  ByVal Message As String)

#If Tracing Then
  Call DebugPrint(Source, Message)
#End If

End Sub
Public Sub TraceParams(ByVal Source As String, _
  ParamArray Values() As Variant)

  Dim Message As String
  Dim I As Integer

  For I = LBound(Values) To UBound(Values)
    Message = Message & " " & Values(I)
  Next I

  Call Trace(Source, Message)

End Sub
#If Debugging Then

Public Sub TrapTest()
 Call Trap("Sheet1.CallTrap", "Test Trap")
End Sub
Public Sub AssertTest()
   Call Assert(False, "AssertTest", "Assertion Failure")
End Sub
Public Sub TraceTest()
   Call Trace("Sheet1.TraceAssert", "Trace Test")
End Sub
Public Sub TestSuite()
   ' Закомментировать после завершения всех тестов
   TrapTest
   AssertTest
```

```
   TraceTest
End Sub
#End If
```

Тестовый код доступен только при установке переменной Debugging в значение True. Метод TestSuite вызывает тестовые методы (TrapTest, AssertTest и TraceTest), которые, в свою очередь, вызывают отладочные методы, а результат вызова можно пронаблюдать в окне Immediate (Проверка). Построчное выполнение кода позволяет убедиться в получении необходимых результатов перед передачей кода другим разработчикам.

В создании обвязки нет ничего сложного. Но это необходимо делать в процессе написания кода. Тестируйте каждый уровень в процессе увеличения сложности, а не все сразу после того, как написание основного кода завершено. Создание нескольких уровней тестов является настолько же важным, как и создание нескольких уровней кода. Каждый фрагмент будет базироваться на надежной основе.

Запись в журнал событий

Журнал событий является системным ресурсом. Он достаточно важен и ценен, чтобы быть неотъемлемой частью новой инфраструктуры .NET от компании Microsoft. (Инфраструктура .NET предназначена для использования в языках программирования Visual Basic, C#, C++ и множестве других языков программирования.) Это настолько важный ресурс, что компания Microsoft включила его в Exception Management Application Block (EMAB) и Enterprise Instrumentation Framework (EIF). Дополнительная информация о EMAB и EIF доступна в сети Internet. В отличие от журнала событий эти механизмы не доступны для непосредственного использования, поэтому ниже рассматривается только журнал событий.

Журнал событий является локальной системной службой, выступающей в роли хранилища информации о состоянии приложений, безопасности и компьютере в целом. Журнал событий можно использовать в процессе диагностики, так как его содержимое сохраняется между сеансами работы приложения. Следовательно, сообщения в журнале событий не исчезнут даже в случае аварийного отказа приложения. Эта информация позволит определить причину отказа. Другими словами, окно Immediate (Проверка) — это хорошо, но журнал событий доступен всегда.

Следующий код доступен в файле EventLog.bas. В коде используется шесть методов Windows API, позволяющих подключиться к журналу событий Windows и упрощающих запись сообщений об ошибках до использования единственного метода WriteEntry. В этом случае не показана вся гибкость возможностей журнала событий, но на данном этапе журнал событий будет использоваться только для записи сообщений об ошибках:

```
Option Explicit
Private Const GMEM_ZEROINIT = &H40 ' Initializes memory to 0

Private Const EVENTLOG_ERROR_TYPE = 1
Private Const EVENTLOG_WARNING_TYPE = 2
Private Const EVENTLOG_INFORMATION_TYPE = 4

Declare Function RegisterEventSource Lib "advapi32.dll" Alias
↳ "RegisterEventSourceA" (ByVal MachineName As String,
↳ ByVal Source As String) As Long

Declare Function ReportEvent Lib "advapi32.dll"
```

```
↳ Alias "ReportEventA" (ByVal Handle As Long,
↳ ByVal EventType As Integer, ByVal Category As Integer,
↳ ByVal EventID As Long, ByVal UserId As Any,
↳ ByVal StringCount As Integer, ByVal DataSize As Long,
↳ Text As Long, RawData As Any) As Boolean

Declare Function DeregisterEventSource Lib "advapi32.dll" (ByVal
↳ Handle As Long) As Long

Declare Sub CopyMemory Lib "kernel32" Alias "RtlMoveMemory" (ByVal
↳ Destination As Any, ByVal Source As Any, ByVal Length As Long)

Declare Function GlobalAlloc Lib "kernel32" (ByVal Flags As Long,
↳ ByVal Length As Long) As Long

Declare Function GlobalFree Lib "kernel32" (ByVal
↳ hMem As Long) As Long

Public Sub WriteEntry(ByVal Message As String)

  Dim Handle As Long
  Dim EventSource As Long

On Error GoTo Finally
  Handle = GlobalAlloc(GMEM_ZEROINIT, Len(Message) + 1)
  Call CopyMemory(Handle, Message, Len(Message) + 1)
  EventSource = OpenEventSource("vbruntime")

  Call ReportEvent(EventSource, EVENTLOG_ERROR_TYPE, _
    0, 1, 0&, 1, 0, Handle, 0)

Finally:
  If (Handle <> 0) Then Call GlobalFree(Handle)
  If (EventSource <> 0) Then CloseEventSource (EventSource)
End Sub
Public Function OpenEventSource(ByVal Source As String) As Long
  ' Использовать локальный компьютер
  OpenEventSource = RegisterEventSource(".", Source)
End Function

Public Sub CloseEventSource(ByVal EventSource As Long)
  Call DeregisterEventSource(EventSource)
End Sub
Sub LogEventTest()

  Call WriteEntry("Это тест!")
End Sub
```

После добавления модуля EventLog можно изменить значение переменной UseEventLog на True и вызывать метод WriteEntry для записи информации об ошибках в журнал событий. Для просмотра журнала приложений воспользуйтесь утилитой Просмотр событий (Event Viewer) (выберите команду Пуск⇨Выполнить (Start⇨Run) и введите Eventvwr.msc). Записи в журнале будут иметь источник vbruntime, как показано на рис. 7.6.

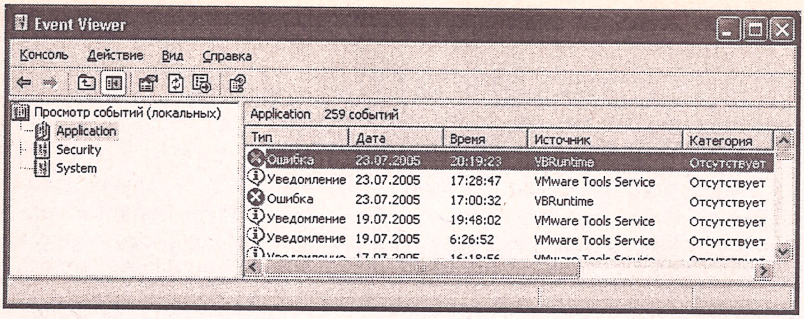


Рис. 7.6. Журнал событий. Записи об ошибках

Модуль `EventLog.bas` можно рассматривать как приманку для главы 16. (Невозможно уместить все в одной главе, поэтому использование Windows API рассматривается более подробно в главе 16, “Программирование с помощью Windows API”.)

Резюме

Для того чтобы что-то сделать хорошо, необходимо просто упорядочить хаос. Для того чтобы что-то сделать хорошо и быстро, необходима практика, привычка и хороший набор инструментов. В этой главе было показано, как создавать полезные отладочные инструменты и инструменты тестирования на основе объекта `Debug`, оператора `Stop` и журнала событий Windows. Выработанная привычка создавать код с помощью таких отладочных инструментов позволяет сразу получать качественный код и делать это быстро.

В этой главе рассматривались методы `Assert`, `Trace` и `Trap`. Кроме этого, было показано, как использовать функции Windows API для записи ошибок в журнал событий. Большинство неясных моментов применения Windows API будут рассмотрены в главе 16.

8

Отладка и тестирование

В продажу выпущены сотни книг по программированию. Интересно, но только небольшая часть из них посвящена тестированию и отладке или содержит главы, посвященные этой теме. При этом во множестве отчетов (что подтверждается опытом) указано, что больше половины времени разработчик тратит на отладку кода. Учитывая такое соотношение, должно существовать больше книг, посвященных отладке и тестированию, и не так много книг, посвященных написанию кода.

Существует несколько основных принципов создания защищенного кода. Первое правило заключается в том, что если код не нужно писать, его не нужно и тестировать, то есть, чем меньше, тем лучше. Второе правило в том, что большая часть времени должна уделяться "исправлению" написанного, а не "написанию" того, что потом придется исправлять. Третье правило подразумевает дополнение кода инструментарием в процессе создания (создание самодиагностирующегося кода), так как в процессе написания разработчик лучше представляет себе все предположения, чем через полгода после завершения проекта. (Скорее всего, через полгода с кодом придется бороться кому-то другому, а самому разработчику — иметь дело с новым кодом. Звучит неплохо, но отладочный инструментарий может отсутствовать и в новом коде.)

Способность искать собственные и, что важнее, чужие ошибки является одной из форм искусства, как игра на музыкальном инструменте или полеты на самолетах. Некоторые обладают механическим профессионализмом, а некоторые демонстируют красоту и изящество. (Если потратить 30 часов на отладку чьего-то кода на C++, а отладка кода другого разработчика займет только минуту, код второго разработчика можно считать примером красоты и изящества.) В этой главе будут показаны инструменты, позволяющие получить механический профессионализм в отладке и тестировании. Красота и изящество придут только с практикой.

Пошаговое выполнение кода

Существует несколько методов тестирования. Терминология может различаться, но обычно применяются такие термины, как модульное тестирование (unit testing), тестирование белого ящика (whitebox testing), тестирование серого ящика (grey box testing), тестирование черного ящика (black box testing), регрессионное тестирование (regression testing), интеграционное тестирование (integration testing), тестирование приложения (application testing) и предприятия (enterprise testing). Каждый термин имеет определенное значение. С точки зрения разработчика существует два принципа: нельзя быть уверенным в ожидаемом поведении кода, если не проверен каждый маршрут выполнения, и нельзя быть уверенным в ожидаемом поведении кода. Возможно, лучшим подходом является написание минимального объема кода. В большинстве ситуаций код будет вести себя ожидаемым образом, а если что-то пойдет не так, отказ не будет катастрофическим.

Потребители будут не очень довольны, если приложение потеряет результаты нескольких часов работы, поэтому для снижения вероятности такого исхода необходимо планомерно просмотреть весь код и избавиться от как можно большей части дефектов. К сожалению, отладка и тестирование могут оказаться исключительно сложными процессами. Эту сложность практически невозможно донести до начинающих разработчиков. Точно так же невозможно смоделировать один из сложных сценариев в этой книге. Но здесь можно показать механизмы пошагового просмотра кода, а профессионализм придет со временем.

Выполнение кода

Роберт Фрост (Robert Frost) сказал: "Разница заключается в том, что я выбрал нехоженную дорогу". Это относится и к программному обеспечению. Если методично пройти по всем маршрутам кода, включая редко используемые, конечный продукт на основе этого кода будет более жизнеспособным и содержащим меньше сюрпризов для потребителя. Первым шагом на нехоженной дороге является запуск кода перед передачей потребителю.

В Excel VBA для запуска кода можно воспользоваться меню Run (Выполнить) в редакторе VBE. Кроме этого, в редакторе VBE можно нажать клавишу <F5>, после чего также начнется выполнение кода. Хорошей практикой является создание парной тестовой подпрограммы, которая не требует передачи аргументов и предназначена для вызова метода. Этот подход называется созданием обвязки. Если создать подпрограмму, вызывающую каждый или, как минимум, большинство методов, их можно будет тестировать по отдельности. Этот подход позволяет значительно ускорить процесс тестирования. В следующем листинге показан метод, который может потребоваться для решения проблемы, и тестовая подпрограмма, вызывающая метод и предоставляющая удобную точку входа для механизма тестирования.

```
Private Sub TestFahrenheitToCelsius()
  Debug.Print FahrenheitToCelsius(32)

  If (FahrenheitToCelsius(32) <> 0) Then
    Debug.Print "TestFahrenheitToCelsius: Неудача"
    Debug.Assert False
  Else
    Debug.Print "TestFahrenheitToCelsius: Успех"
  End If
```

```
End Sub

Public Function FahrenheitToCelsius(ByVal _
TemperatureFahrenheit As Double) As Double

  FahrenheitToCelsius = (5 / 9 * (TemperatureFahrenheit - 32))

End Function
```

В этом примере используется открытая функция, выполняющая преобразование показателей температуры по Фаренгейту в показатели по Цельсию. Для проверки правильности уравнения можно написать тестовый метод, который будет проверять преобразование. Зная температуру кипения и замерзания (212, 100 и 32, 0 соответственно), алгоритм преобразования можно проверить с помощью закрытого метода, который сравнивает ожидаемые результаты с известными аргументами.

В данном примере `FahrenheitToCelsius` описывает необходимый алгоритм, а `TestFahrenheitToCelsius` является закрытым методом, используемым для тестирования независимо от остальных компонентов. Если тестовый метод успешно выполнил свою функцию, в списке методов можно поставить пометку о правильности работы основного метода. То есть, дальнейшая проверка не требуется. Если каждый метод отмечен, как надежный, то приложение должно работать правильно при любых входных параметрах.

В реальности обвязка используется для исключения большого количества ошибок, но небольшие потенциальные ошибки могут накапливаться и приводить к реальным ошибкам, возможность появления которых никогда не тестировалась. Например, а что, если пользователь введет число, которое выходит за пределы допустимого диапазона значений типа `Double`? В таком случае появится сообщение о неожиданной ошибке и даже протестированный код будет отказывать. Другими словами, тщательное тестирование позволяет избавиться от большинства, но не от всех ошибок.

Только разработчик и потребители могут принять решение о допустимом проценте ошибок. Если программное обеспечение используется для рассчета налога на прибыль, то могут быть допустимы минимальные ошибки в 5 из 100 случаев. Если программное обеспечение определяет момент срабатывания тактового генератора, то допустимый уровень ошибок может составлять 1 из 10000000. Так как в данном случае обсуждается программирование приложений для Excel, читатели, вероятнее всего, будут подсчитывать ставки налогов, а не создавать встроенное программное обеспечение для тактовых генераторов. Но в любом случае важно, чтобы разработчик и потребитель оговорили среднее время между ошибками, так как полное отсутствие отказов при нынешнем уровне технологии невозможно.

Шаг с заходом

Команда Step Into (Шаг с заходом) доступна в меню Debug (Отладка) редактора VBE. Для вызова этой команды можно воспользоваться клавишей <F8>. Команда Step Into (Шаг с заходом) поочередно выполняет каждую строку кода. Если нажать клавишу <F8> на строке вызова метода, отладчик перейдет к данному методу. Применение этой команды требует значительных трудозатрат, но позволяет проверить каждую строку кода.

При использовании этой команды разработчик видит, где и какой код будет выполняться следующим. Для демонстрации применения команды откройте редактор VBE и наведите курсор на метод `TestFahrenheitToCelsius`. Нажмите клавишу <F8>. Строка

кода, которая будет выполняться следующей, выделяется ярко-желтым фоном. На рис. 8.1 выделен заголовок подпрограммы. Еще раз нажмите клавишу <F8>, и выделение переместится на строку вызова `Debug.Print`. Нажмите <F8> в третий раз, и отладчик перейдет к коду метода `FahrenheitToCelsius` и т.д.

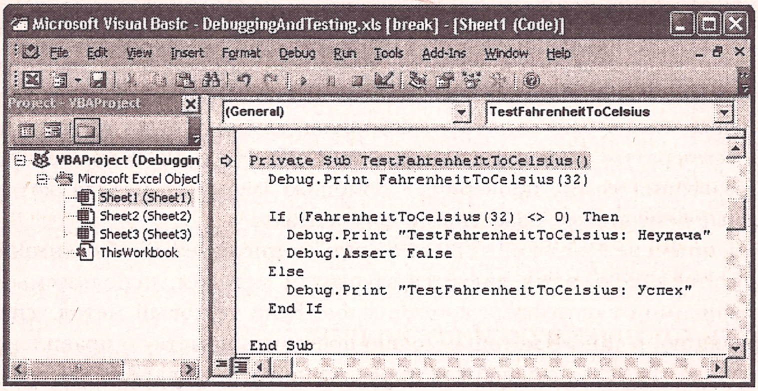


Рис. 8.1. Использование команды Шаг с заходом

Шаг с обходом

Комбинация клавиш <Shift+F8> вызывает команду Debug⇨Step Over (Отладка⇨Шаг с обходом) в редакторе VBE. Например, если после метода `Debug.Print` не нужно переходить внутрь процедуры `FahrenheitToCelsius`, нажмите комбинацию клавиш <Shift+F8>. Метод будет вызван, но отладчик не перейдет к строкам кода этого метода.

Шаг с выходом

Если проверяется выполнение длинного метода или метода, который уже считается надежным, воспользуйтесь комбинацией клавиш <Ctrl+Shift+F8> для запуска команды Step Out (Шаг с выходом). (Команда Step Out (Шаг с выходом) доступна в меню Debug (Отладка) в редакторе VBE.) Команда Step Out (Шаг с выходом) позволяет выполнить оставшийся код в процедуре и сразу выйти за пределы текущей области видимости.

Предположим, что при пошаговом выполнении метода определен и исправлен источник ошибки. Вместо ручного выполнения оставшихся строк метода можно воспользоваться комбинацией клавиш <Ctrl+Shift+F8> и выполнить метод до конца без вмешательства разработчика. Кроме этого, в любой момент можно нажать клавишу <F5> и перейти в режим нормального выполнения.

Выполнить до текущей позиции

Кроме шага с заходом, обходом и выходом, можно щелкнуть на любой строке кода и воспользоваться командой Debug⇨Run to cursor (Отладка⇨Выполнить до текущей позиции) или нажать комбинацию клавиш <Ctrl+F8>. Отладчик будет выполнять код, пока не достигнет указанной строки. После этого выполнение будет приостановлено. Это подходящий прием при исправлении фрагмента кода для автоматического выполнения заведомо исправленных строк.

Следующая инструкция

Команда Debug⇨Set Next Statement (Отладка⇨Следующая инструкция) доступна с помощью комбинации клавиш <Ctrl+F9> и позволяет замкнуть строки кода в пределах одной процедуры. Еще раз обратимся к подпрограмме `TestFahrenheitToCelsius`. Предположим, что необходимо проверить правильность вывода передаваемого сообщения с помощью метода `Debug.Print` в блоке `Else`. Метод `TestFahrenheitToCelsius` можно запустить с помощью клавиши <F8>. После этого щелкнуть на вызове `Debug.Print` в блоке `Else` и нажать комбинацию клавиш <Ctrl+F9>. Все предыдущие строки кода будут пропущены и отладчик выполнит код в указанной строке.

Эта возможность особенно полезна, так как дает шанс пропускать строки кода, которые вносят критические модификации до того, как подготовка к этим модификациям полностью завершена. После выполнения кода создания и удаления обвязки можно пропустить модификацию критических фрагментов данных и проверить работоспособность процедуры без этой модификации.

Отобразить следующую инструкцию

В процессе отладки часто возникает необходимость в остановке кода и поиске необходимой информации, например, при отладке процедуры может потребоваться информация о поле класса. При этом можно потерять текущее положение отладчика. Выберите команду Debug⇨Show Next Statement (Отладка⇨Отобразить следующую инструкцию), и редактор VBE автоматически перенесет курсор на строку кода инструкции, которая будет выполняться следующей.

Все вместе команды: Step Into (Шаг с заходом), Step Over (Шаг с обходом), Step Out (Шаг с выходом), Run to Cursor (Выполнить до текущей позиции), Set Next Statement (Следующая инструкция) и Show Next Statement (Показать следующую инструкцию) позволяют перемещаться по коду и избавляют от тысяч нажатий комбинаций клавиш в процессе отладки.

Использование точек останова

В главе 7 кратко рассматривалась история отладчиков и точек останова. В этой главе новое понимание точек останова используется для разделения и победы над проблемами. Для установки точки останова необходимо выбрать интересующую строку кода и нажать клавишу <F9> (или выбрать команду меню Debug⇨Toggle Breakpoint (Отладка⇨Точка останова). Точка останова обозначается красным кругом в начале строки. В процессе отладки можно установить несколько десятков таких точек. Для быстрого удаления всех точек останова после завершения отладки выберите команду Debug⇨Clear All Breakpoints (Отладка⇨Снять все точки останова). Эта команда также доступна в виде комбинации клавиш <Ctrl+Shift+F9>.

Теперь, когда известен механизм работы точек останова в редакторе VBE, их можно применять для решения поставленных задач. Для этого можно воспользоваться известным изречением Divide et impera (Разделяй и властвуй). Самым быстрым способом обнаружения ошибки, расположение которой заранее неизвестно, особенно в незнакомом коде, является установка точки останова и запуск кода. Если точка останова встретилась до появления ошибки, то ошибка расположена в коде после нее. Добавьте вторую точку останова во второй половине кода. Если вторая точка достигнута до появления ошибки, установите еще одну точку останова после второй точки. Если и в этот раз ошибка не про-

явилась, добавьте точку останова после третьей точки останова. Если ошибка появилась после первой точки останова, но перед второй, добавьте точку останова между ними. Повторение этой процедуры позволит быстро найти ошибку даже в совершенно незнакомом коде. (Этот подход основан на логарифме по основанию 2 и еще называется методом дихотомии или "Разделяй и властвуй".)

Многие простые ошибки проявляются в тех местах кода, где они находятся. Такие ошибки исправляются очень просто. Ошибки, для обнаружения которых требуется дихотомическое размещение точек останова, могут занять большую часть рабочего времени программиста, но использование точек останова значительно помогает в изолировании таких проблем.

Использование контрольных значений

Утверждение "Знание — сила" хорошо подходит разработчикам программного обеспечения. Чем больше профессиональных знаний и информации о текущем состоянии программного обеспечения, тем более мощными, выразительными и полезными будут решения разработчиков. Мы программируем достаточно долго, чтобы помнить создание кода в простых текстовых редакторах, запуск компиляции из командной строки и реализацию стратегий отладки в манере "ловить все, что ловится". С этой точки зрения очень подходят современные интегрированные инструменты разработки, например VBE, которые появились относительно недавно (в течение последнего десятка лет).

В меню Debug (Отладка) доступны команды Add Watch (Добавить контрольное значение), Edit Watch (Изменить контрольное значение) и Quick Watch (Контрольное значение) (последняя команда может запускаться с помощью комбинации клавиш <Shift+F9>. Для использования контрольных значений необходимо выбрать интересующую переменную, объект или выражение, и выполнить соответствующую команду. С другой стороны, можно выполнить команду и после этого выбрать интересующую переменную, объект или выражение.

Добавление контрольного значения

Команда Debug⇨Add Watch (Отладка⇨Добавить контрольное значение) позволяет добавить переменную, объект или выражение в немодальное окно. Окно Watch (Контрольное значение) выводится при работе кода в отладочном режиме. Контрольные значения в пределах текущей области видимости обновляются в процессе работы программы.

Для демонстрации работы этих команд рассмотрим метод, который вызывает метод `FahrenheitToCelsius`, передавая параметры от 1 до 400. Вместо ручной проверки вывода после каждого преобразования можно добавить контрольное значение, показывающее количество градусов по Фаренгейту (номер итерации цикла) и результат вызова функции. В следующем листинге показан используемый код:

```
Private Sub TestWatch()

  Dim I As Integer
  For I = 1 To 400
    Debug.Print FahrenheitToCelsius(I)
  Next I

End Sub
```

Для добавления контрольного значения на основе выражения `FahrenheitToCelsius(I)` можно выделить выражение в редакторе и выбрать команду Debug⇨Add Watch (Отладка⇨Добавить контрольное значение). Появится диалоговое окно Add Watch (Добавить контрольное значение) (рис. 8.2), в котором показано выражение (в данном случае, переменная, объект или выражение), контекст, состоящий из процедуры и модуля, и тип контрольного значения. Принятый по умолчанию тип контрольного значения просто позволяет следить за текущим состоянием выражения. Кроме этого, предоставляется возможность остановки выполнения программы при изменении контрольного значения или при истинности условия на его основе. Условные контрольные значения позволяют выполнять код без остановок до выполнения определенного условия.

После завершения настройки контрольного значения щелкните на кнопке OK. После закрытия диалогового окна Add Watch (Добавить контрольное значение) откроется диалоговое окно Watch (Контрольные значения), в котором появится выбранное выражение. Здесь рассматриваются два контрольных значения: итератор цикла (`I`) и вызов метода (рис. 8.3). При входе в метод `TestWatch` в области видимости появляются два выражения, и окно Watch (Контрольные значения) начинает постоянно обновляться.

Кроме этого, диалоговое окно Watch (Контрольные значения) предоставляет возможность просмотра содержимого сложных объектов. Например, если в окно Watch (Контрольные значения) добавить объект (рис. 8.4), можно щелкнуть на символе [+] и просмотреть текущее состояние интересующего объекта. Как и другие выражения, контролируемое состояние объекта обновляется в момент изменения состояния. На рис. 8.4 показана часть внутреннего состояния объекта листа `Sheet1`.

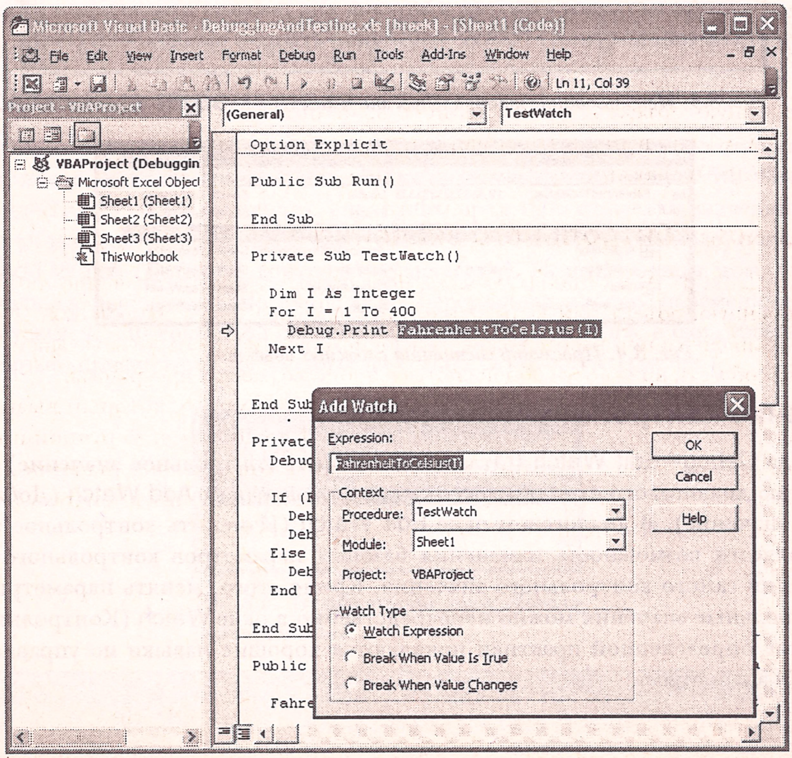


Рис. 8.2. Добавление контрольного значения

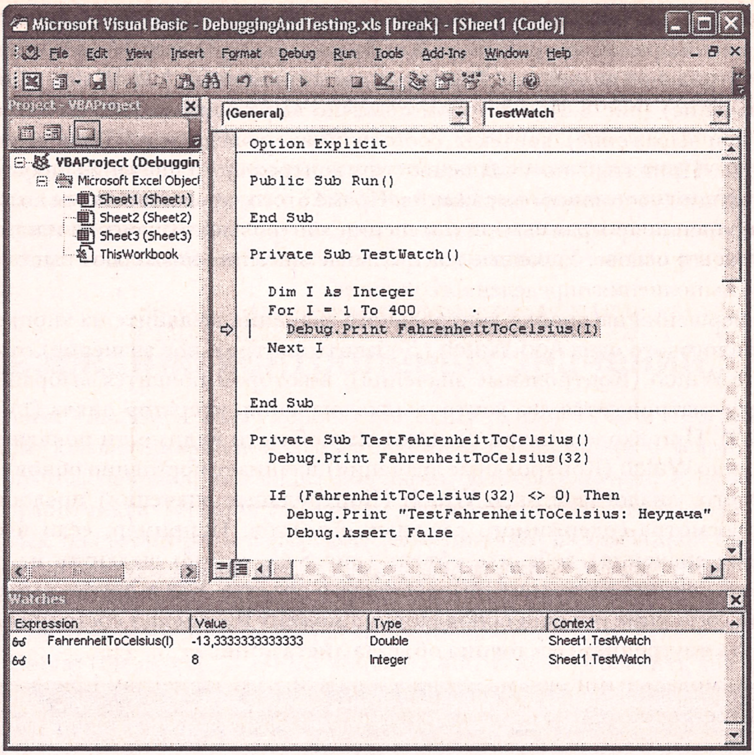


Рис. 8.3. Использование контрольных значений

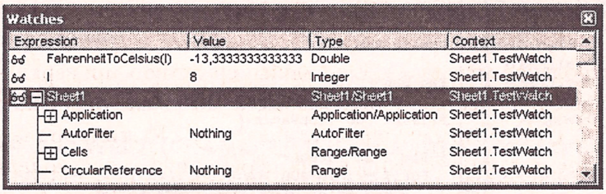


Рис. 8.4. Просмотр состояния сложных объектов

Изменение контрольного значения

Команда Debug⇨Edit Watch (Отладка⇨Изменить контрольное значение) приводит к появлению диалогового окна, похожего на диалоговое окно Add Watch (Добавить контрольное значение). В диалоговом окне Edit Watch (Изменить контрольное значение) предоставляется возможность изменения базовых параметров контрольного значения или удаления самого контрольного значения. Кроме этого, менять параметры или удалять контрольные значения можно непосредственно в окне Watch (Контрольные значения). После определенной практики появляются хорошие навыки по управлению контрольными значениями.

Контрольное значение

Команда Debug⇨Quick Watch (Отладка⇨Контрольное значение) доступна из меню редактора VBE или с помощью нажатия комбинации клавиш <Shift+F9>. Диалоговое окно Quick Watch (Контрольное значение) является модальным. Для его использования выделите интересующее выражение и нажмите комбинацию клавиш <Shift+F9>. Как показано на рис. 8.5, в диалоговом окне Quick Watch (Контрольное значение) отображается выражение и значение выражения. Если выражение необходимо добавить в диалоговое окно Watch (Контрольные значения), щелкните на кнопке Add (Добавить).

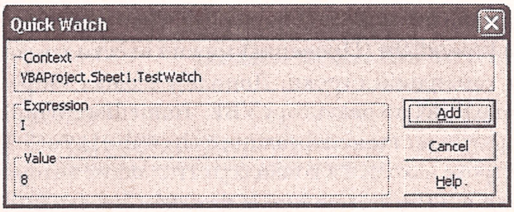


Рис. 8.5. Использование диалогового окна Контрольное значение

Так как диалоговое окно Quick Watch (Контрольное значение) является модальным, во время его вывода на экран выполнение программы приостанавливается. Как и в случае с прочими модальными окнами, для доступа к другим элементам приложения (в данном случае, Excel) необходимо закрыть окно Quick Watch (Контрольное значение).

Окно Локальные переменные

Если случайно закрыть окно Watch (Контрольные значения), его можно открыть повторно, добавив новое контрольное значение или выбрав в меню редактора VBE View⇨Watch Window (Вид⇨Контрольное значение). Еще одним средством проверки значений является окно Locals (Локальные переменные). Окно Locals (Локальные переменные) очень напоминает окно Add Watch (Добавить контрольное значение). Разница лишь в том, что в окне Locals (Локальные переменные) показаны не все переменные, а только переменные, которые доступны в текущей или локальной области видимости. Кроме этого, в окне Locals (Локальные переменные) присутствует доступная везде ссылка на объект Me (рис. 8.6).

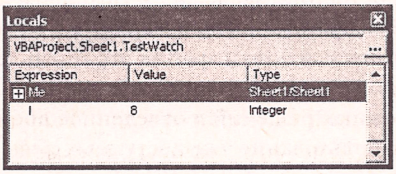


Рис. 8.6. Использование окна Локальные переменные

> Переменная Me является внутренней ссылкой объекта на самого себя. Ее можно использовать в качестве удобного способа вывода окна Members (Члены). Это же окно открывается вручную, если нажать комбинацию клавиш <Ctrl+пробел> в редакторе VBE.

Тестирование выражения в окне Проверка

Окно Immediate (Проверка) представляет собой интерпретатор внутри редактора. Для доступа к окну выберите команду меню View⇨Immediate (Вид⇨Проверка) или нажмите комбинацию клавиш <Ctrl+G>. В этом окне вводятся команды, переменные, объекты и операторы, которые выполняются немедленно. В результате программист может проверять предположения и альтернативные варианты, не создавая дополнительный код.

Чаще всего окно используется для вывода значения переменной из текущего контекста, для этого применяется команда `print` или символ `?`. После команды `print` можно вызвать практически любую функцию, например, `? abs(-5)`. Результат 5 будет выведен на следующей строке в окне Immediate (Проверка). Если память о командной строке DOS все еще сильна, в окне Immediate (Проверка) можно ввести команду **`cmd`** и получить доступ к интерпретатору командной строки. Дополнительная информация доступна в разделе справочного руководства по редактору VBE "Immediate Window Keyboard Shortcuts".

На рис. 8.7 показан результат вызова функции `FahrenheitToCelsius` из окна Immediate (Проверка). В первой строке находится команда `print`, после которой указан вызов функции и ее результат. Обратите внимание, что вызов функции необходимо квалифицировать именем экземпляра класса, в котором она определена. Если функция определена в модуле, квалификация не требуется. Так как функция `FahrenheitToCelsius` определена в листе `Sheet1`, придется воспользоваться квалификатором `Sheet1` и оператором членства (`.`).

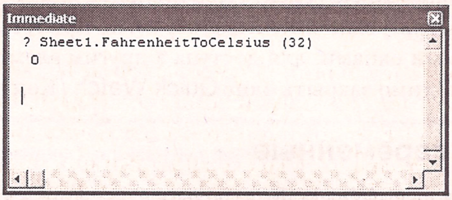


Рис. 8.7. Вызов функции FahrenheitToCelsius из окна Проверка

Источники получения информации об определениях

До этого момента было продемонстрировано, как просматривать код и получать информацию о его текущем состоянии. Надежным правилом является создание тестового кода, который проверяет каждый возможный маршрут и максимальное количество вариантов результатов. С практической точки зрения это очень сложно и долго; по этой причине повторное использование максимального объема кода является лучшим способом разработки и создания надежных решений в отведенное время. Следующий фрагмент головоломки — это поиск и использование уже доступных решений. Рассмотрим возможности редактора VBE, помогающие в реализации этого правила.

Команда Краткие сведения

Команда Quick Info (Краткие сведения) (<Ctrl+I>) приведет к появлению всплывающей подсказки об элементе, который находится под курсором. Например, если навести курсор на слово `FahrenheitToCelsius` и нажать комбинацию клавиш <Ctrl+I>, сигнатура метода будет показана в виде всплывающей подсказки (рис. 8.8).

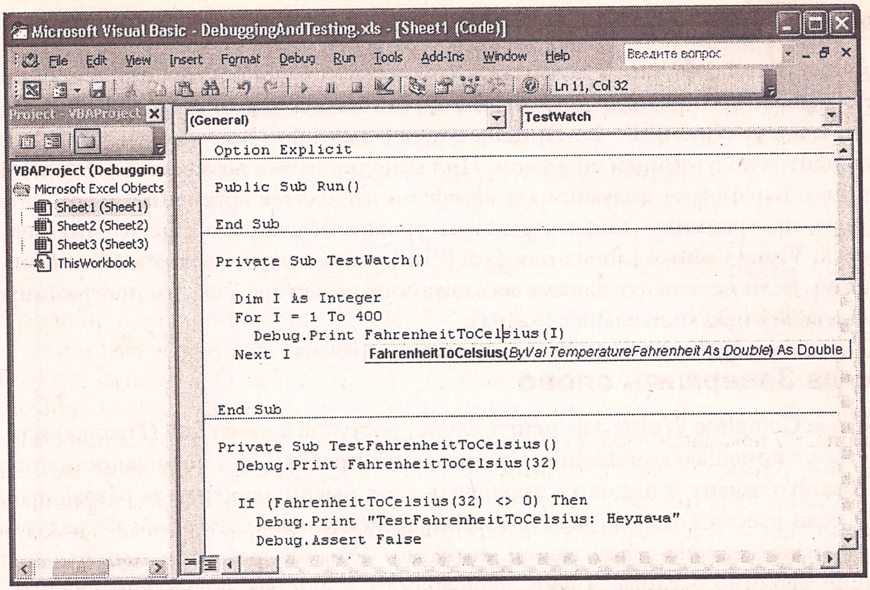


Рис. 8.8. Использование команды Краткие сведения для получения информации об элементе кода

Подсказки с краткими сведениями по умолчанию включаются в меню Tools⇨Options (Сервис⇨Параметры) на вкладке Editor (Редактор) (рис. 8.9). Но новички в Excel или пользователи редактора VBE, в котором кто-то отключил автоматическое отображение кратких сведений, могут столкнуться со сложностями при получении информации об использовании определенных возможностей Excel. (В Excel существует слишком много классов, методов, свойств, событий и полей, чтобы один человек мог запомнить информацию, предоставляемую в подсказках Quick Info (Краткие сведения).

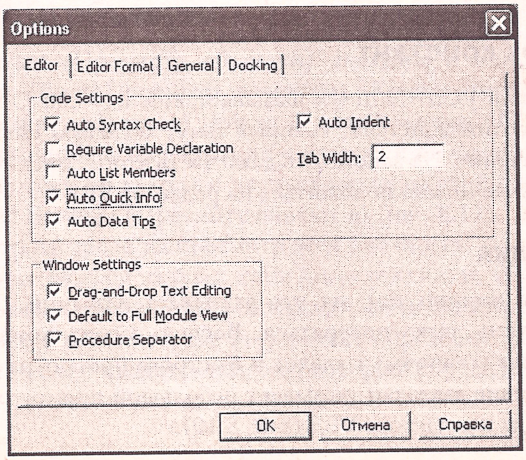


Рис. 8.9. Включение подсказок с краткими сведениями

Команда Сведения о параметре

Команда Parameter Info (Сведения о параметре) предоставляет информацию об аргументах выбранного метода. Для доступа к этой возможности необходимо нажать комбинацию клавиш <Ctrl+Shift+I>. При доступе к Quick Info (Кратким сведениям) имя метода выводится полужирным шрифтом. При использовании возможности Parameter Info (Сведения о параметре) полужирным шрифтом выводятся аргументы. Кроме этого, современные программные инфраструктуры уровня Microsoft Office (например, .NET Framework, Visual Control Library или Java J2EE) оказываются слишком большими для запоминания. Если не освоить данные возможности редактора VBE, то программирование может оказаться исключительно сложным.

Команда Завершить слово

Команда Complete Word (Завершить слово) доступна в меню Edit (Правка) или может вызываться с помощью комбинации клавиш <Ctrl+пробел>. Эта возможность просматривает текущий элемент и пытается завершить ввод имени элемента за разработчика. Например, если ввести `Fahre` и нажать комбинацию клавиш <Ctrl+пробел>, редактор VBE автоматически закончит ввод имени функции. Обычно редактор неплохо угадывает окончание вводимого слова. Такие возможности помогают правильно вводить имена сущностей, которые сложно ввести самостоятельно, например `Fahrenheit`.

Команда Список свойств/методов

Команда List Properties/Methods (Список свойств/методов) выводит свойства и методы, доступные в текущем контексте. Например, если ввести имя объекта, после которого указать оператор членства, в раскрывающемся списке будут показаны свойства и методы этого объекта. По умолчанию эта возможность есть в диалоговом окне Tools⇨Options (Сервис⇨Параметры) на вкладке Editor (Редактор) (установлен флажок Auto List Member). Если сбросить этот флажок, то для вывода раскрывающегося списка свойств и методов придется нажать комбинацию клавиш <Ctrl+J>.

Команда Список констант

Комбинация клавиш <Ctrl+Shift+J> вызывает команду List Constants (Список констант). Существуют сотни констант, которые несут большую смысловую нагрузку, чем простое числовое значение. Но запомнить даже часть этого списка — непосильная задача. Поэтому в данном случае можно положиться на редактор VBE.

Команда Закладка

Закладка является именно тем, на что указывает название. При перемещении по большому проекту очень легко потеряться. Воспользуйтесь командой Edit⇨Bookmark (Правка⇨Закладка) для установки закладок и быстрого просмотра списка существующих закладок. На присутствие закладки указывает небольшой прямоугольник со скругленными углами слева от кода в редакторе VBE (рис. 8.10).

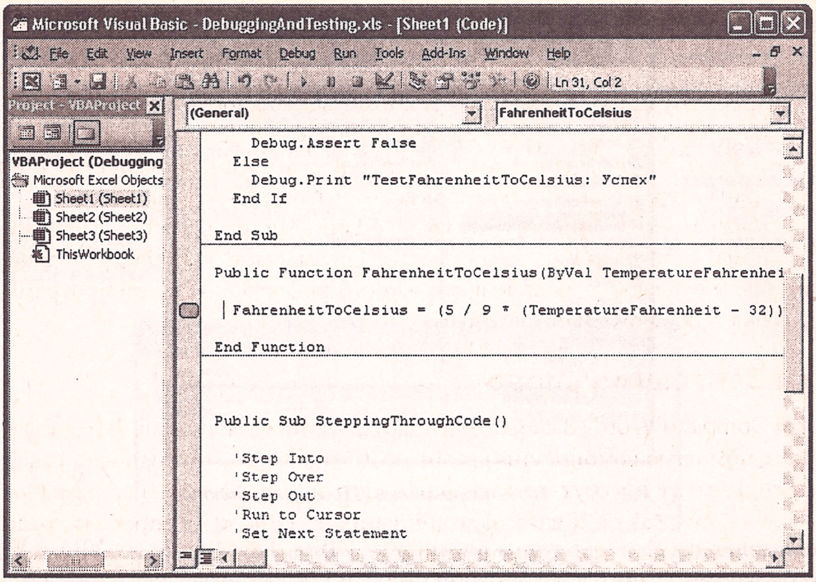


Рис. 8.10. Установка закладки

Команда Описания

Пункт Definition (Описания) позволяет перенести фокус редактора к определению символа. Например, в тестовом приложении можно быстро перейти к определению метода, если навести курсор на символ и вызывать команду меню View⇨Definition (Вид⇨ Описания). Кроме этого, можно нажать комбинацию клавиш <Shift+F2>. В небольшом проекте эта возможность может показаться лишней, но при просмотре большой программы или чужого кода она может пригодиться.

Команда Просмотр объектов

Окно Object Browser (Просмотр объектов) доступно с помощью команды меню View⇨Object Browser (Вид⇨Просмотр объектов). Кроме этого, для доступа к окну Object Browser (Просмотр объектов) можно нажать клавишу <F2>. Это окно является централизованным источником информации о классах и их членах. Если добавить ссылку на внешнюю библиотеку, то классы и члены классов из этой библиотеки также будут доступны в окне Object Browser (Просмотр объектов) (рис. 8.11).

На рис. 8.11 фильтр настроен на просмотр всех библиотек, ссылки на которые добавлены в проект. В списке классов выбран класс `AnswerWizard`. Открытые члены класса `AnswerWizard` показаны в списке справа. В нижней части окна приводится информация об отображаемом объекте и о месте, где этот объект определен. Например, на рис. 8.11 показано, что `AnswerWizard` является классом, который определен в библиотеке Office.

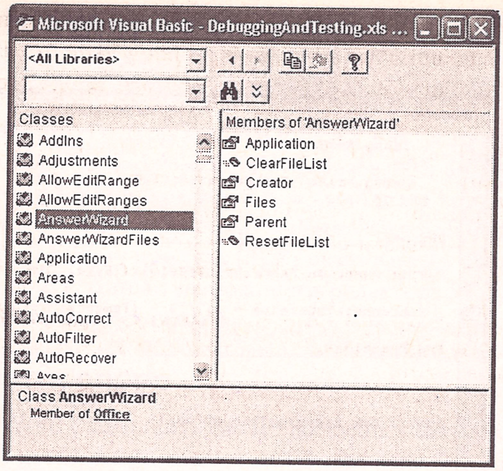


Рис. 8.11. Использование окна Просмотр объектов

По умолчанию для Excel доступны библиотеки Office, Excel, `stdole`, VBA и VBAProject. Каждая из библиотек хранится в отдельном файле и предоставляет редактору VBE собственные возможности. Например, библиотека `stdole` предоставляет доступ к механизму OLE Automation. Эта библиотека находится в файле `C:\WINDOWS\System32\stdole2.tlb`. (В файлах `.tlb` хранятся библиотеки типов, определяющие содержимое объектов COM. Дополнительная информация о механизмах COM и Automation приводится в главе 13.)

Просмотр стека вызовов

Иногда можно встретить фанатика определенного языка программирования. Такие люди считают, что по-настоящему можно программировать только на некоторых языках. (Обычно этим недугом страдают программисты на C++ и Java.) На самом деле ввод текста и успешная компиляция программы, решающей требуемые задачи, является программированием. При этом отдельные инструменты универсальны и могут использоваться для решения практически любой задачи, а некоторые инструменты могут применяться для решения задач только в ограниченной проблемной области. Язык C++ используется для решения любых задач, но на его освоение может потребоваться значительное время. Язык VBA намного проще в изучении и при этом позволяет решать реальные задачи. Каждый язык имеет собственное применение и свои сильные и слабые стороны.

Многие инструменты программиста обладают общими возможностями. Как сложные, так и простые языки выполняют некоторые простые операции. Каждый инструмент должен предоставлять доступ к информации о таких базовых операциях. С появлением функций (функции были изобретены между 1960 и 1975 годами) в программах стало использоваться ветвление, а некоторые фрагменты кода программы стали храниться в одном месте после отказа от многократного копирования одного и того же кода. В результате программистам потребовались инструменты для отслеживания такого ветвления. Подобный инструмент имеет простой принцип действия: при вызове функции текущий адрес и дополнительная информация о состоянии записываются в область памяти, называемой стеком или стеком вызовов. Эта область памяти используется компьютером для хранения адреса перед точкой ветвления и локальных переменных, которые созда-

ются после ветвления. Перед завершением работы функции локальные переменные удаляются из стека и на его вершине остается адрес точки ветвления. Компьютер использует этот адрес в качестве "хлебных крошек" для возврата и продолжения исполнения кода после точки ветвления.

Для просмотра стека вызовов можно выбрать команду меню View⇨Call Stack (Вид⇨Стек вызовов) или нажать комбинацию клавиш <Ctrl+L>. Эта возможность избавляет от угадывания реальной последовательности выполнения кода центральным процессором. На рис. 8.12 показан обратный порядок вызовов при переходе из метода `TestFahrenheitToCelsius` в метод `FahrenheitToCelsius`.

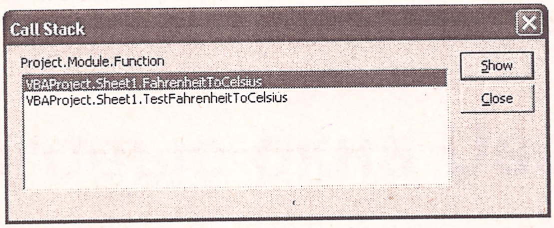


Рис. 8.12. Просмотр порядка вызовов процедур

Проверка инвариантных предположений

В главе 7 рассматривался инструментарий, основанный на методе `Assert` объекта `Debug`, и было показано, как использовать эту возможность языка VBA. Перед завершением этой главы стоит повториться, поговорив о важности проверки предположений в качестве инструмента отладки.

При создании кода разработчик должен убедить себя в правильности решения. Если создаваемый код похож на спагетти, остановитесь и обдумайте решение еще раз. Если при описании реализации другому разработчику она кажется сложной, скорее всего так и есть. Наконец, при создании кода разработчик имеет некоторые предположения о состоянии приложения, входных параметрах и результатах. Вставьте проверку этих предположений в код приложения. В процессе создания кода для каждого такого предположения добавьте вызов метода `Debug.Assert` или воспользуйтесь инструментарием, который рассматривался в главе 7. Через месяц, неделю, день или даже час эти предположения забудутся. Хорошим правилом является составление имен методов из существительного и глагола. Имя должно описывать назначение метода. При этом метод должен состоять из небольшого количества строк кода и содержать проверки предположений. Знаменитый Дэйв Тилен (Dave Thielen) сказал: "Проверьте, что мир существует!" — а уважаемый Джувал Лоуи (Juval Lowy) предложил еще одно правило: "Добавляйте оператор `aAssert` к каждым пяти строкам кода".

Резюме

В результате одного очень дорогого исследования (уже невозможно вспомнить, о чем было это исследование) было получено следующее правило: для творческого мышления в пределах определенной профессии необходимо досконально изучить язык этой профессии. Это совершенно справедливо по отношению к программированию. Разработчик должен изучить грамматические особенности языка для творческого создания решений.

При этом необходимо освоить инфраструктуру и инструменты, которые предоставляются данным языком.

Нет ничего сложного в создании простой функции или пары функций. Но создание целого решения может потребовать значительных трудозатрат. Описанные в главе 7 стратегии и в этой главе инструменты помогут изучить возможности редактора VBE.

Кроме освоения возможностей VBE, языка Visual Basic и инфраструктур VBA, Excel и Office, стоит обратить внимание на шаблоны проектирования и методы рефакторинга кода. Хотя эти темы выходят за пределы рассматриваемых в данной книге вопросов, они могут оказаться очень полезны при решении сложных проблем. Они не нужны при реализации простой функциональности для листов, но превращение нескольких строк VBA в произведение искусства требует совершенно другого подхода.

9

Диалоговые окна UserForm

Строки, столбцы и ячейки предоставляют возможность ввода информации, однако не являются лучшим средством для решения этой задачи. Диалоговые окна UserForm представляют собой основу, позволяющую создавать визуальные метафоры, которые помогают пользователям вводить данные.

Для добавления диалогового окна UserForm (которое будет применяться в качестве основы для визуальных метафор) можно воспользоваться командой меню View⇨UserForm (Вид⇨UserForm). Метафора рисования поддерживается диалоговым окном (холст) и возможностью перетаскивания элементов управления на пустое диалоговое окно (рисование). Доступные элементы управления показаны на панели инструментов Toolbox (Элементы управления). Формы и добавляемые элементы управления имеют определенные методы, свойства и события, позволяющие управлять взаимодействием пользователя и кода. Внешний вид, возможности и реакции диалогового окна ограничиваются только фантазией разработчика.

Отображение диалогового окна UserForm

Принцип работы приложений пакетной обработки сложился исторически и заключается в получении нескольких аргументов, обработка которых приводит к определенному результату. Метафора оконного интерфейса значительно расширила возможности разработчиков, превратив линейную пакетную обработку в динамический массив оппортунистических переходов с целым набором возможных результатов. То есть, оконный интерфейс позволил добиться большей гибкости приложений.

Для использования преимуществ оконной метафоры достаточно добавить в книгу диалоговое окно `UserForm`. В книгу можно добавлять любое количество диалоговых окон. Количество и порядок отображаемых диалоговых окон отражают ограничения и возможности книги как приложения Windows. Управление этими возможностями осу-

ществляется через загрузку, отображение, сокрытие и выгрузку диалоговых окон UserForm в результате действий пользователя.

Чтобы загрузить диалоговое окно UserForm, которое называется UserForm1 (имя, принятое по умолчанию), в память без отображения на экране, можно воспользоваться оператором Load:

```
Load UserForm1
```

Для выгрузки диалогового окна UserForm1 из памяти можно воспользоваться оператором UnLoad:

```
UnLoad UserForm1
```

Для вывода диалогового окна UserForm1 на экран необходимо воспользоваться методом Show объекта UserForm:

```
UserForm1.Show
```

Если вывести на экран диалоговое окно, которое еще не было загружено в память, оно загрузится автоматически. Метод Hide позволяет скрыть диалоговое окно, не выгружая его из памяти, например:

```
UserForm1.Hide
```

На рис. 9.1 показан пример диалогового окна UserForm, которое будет разрабатываться на протяжении этой главы. Данное диалоговое окно предназначено для просмотра и изменения значений в ячейках B2:B6 и непосредственно связано с ячейками листа, поэтому для его настройки требуется минимальный объем кода VBA.

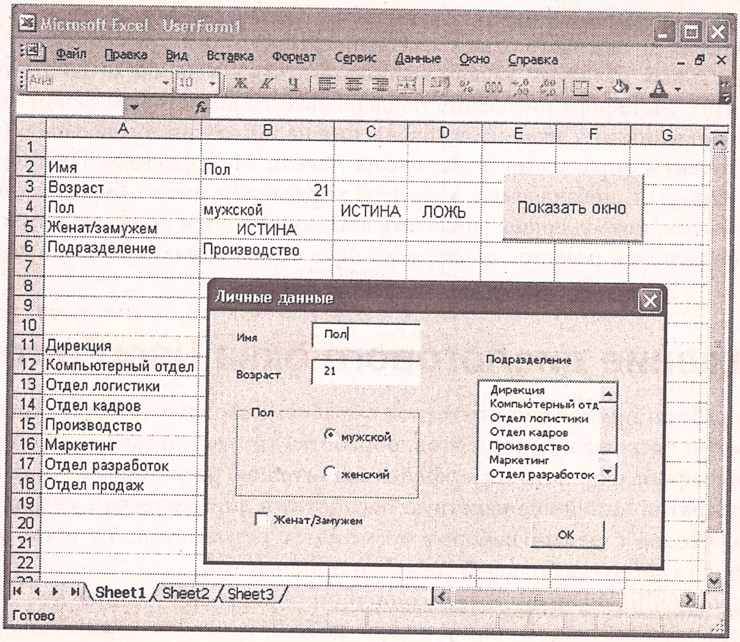


Рис. 9.1. Пример диалогового окна UserForm

На лист добавлена командная кнопка ActiveX с надписью “Показать окно”. С кнопкой связана следующая процедура:

```
Private Sub CommandButton1_Click()
   PersonalData.Show
End Sub
```

По умолчанию при вызове метода `Show` выводится модальное диалоговое окно. Это значит, что пока диалоговое окно `UserForm` не будет скрыто или выгружено из памяти, оно сохранит фокус, из-за чего пользователь не сможет взаимодействовать с другими частями Excel.

В этой главе будут рассматриваться немодальные диалоговые окна `UserForm`*, позволяющие пользователю выполнять другие задачи, даже если диалоговое окно остается на экране.*

Создание диалогового окна UserForm

Диалоговые окна `UserForm` проектируются в редакторе VBE. На рис. 9.2 показана среда проектирования в редакторе VBE в процессе создания диалогового окна UserForm Личные данные.

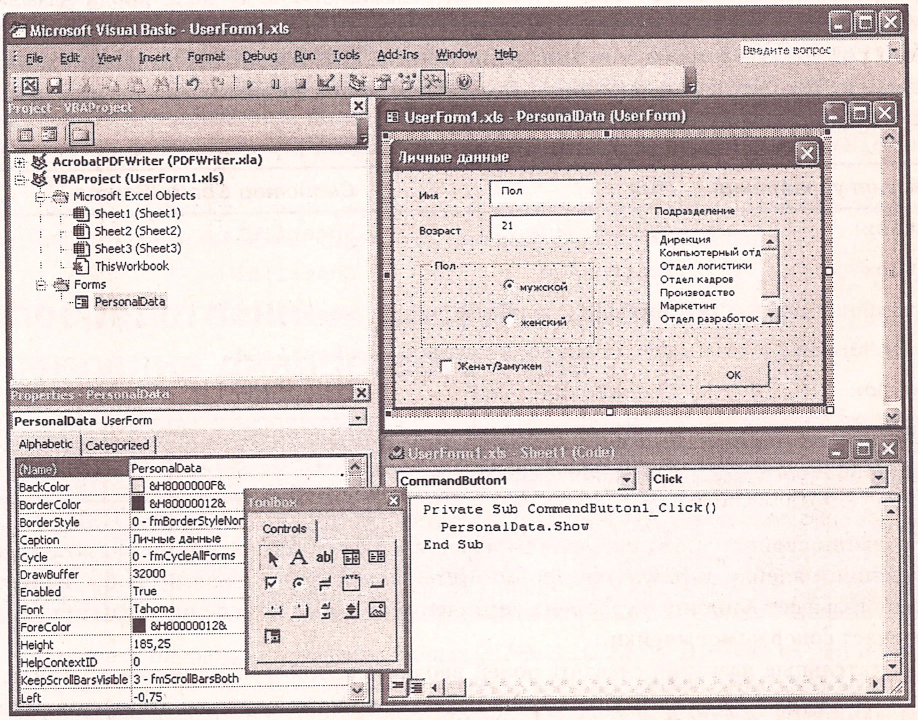


Рис. 9.2. Редактирование диалогового окна UserForm

Принятое по умолчанию имя диалогового окна UserForm1 было изменено на PersonalData. Для этого нужно поменять значение первого свойства (Name) в окне Properties (Свойства). Свойству Caption необходимо присвоить значение **Личные данные**. Элементы управления добавляются из панели ToolBox (Элементы управления).

В верхней части диалогового окна находятся два элемента управления TextBox, которые предназначены для ввода имени и возраста. Кроме этого, доступны два переключателя (Мужской и Женский), заключенные в элемент управления фрейма. Для создания такого фрейма сначала добавьте элемент управления Frame, а потом — элементы управления OptionButtons. Кроме этого, в диалоговом окне PersonalData присутствует флажок CheckBox для указания семейного положения и список ListBox для выбора отдела. Кнопка CommandButton содержит надпись OK.

Имеет смысл использовать описательные имена элементов управления, взаимодействующих с кодом. Например, поле ввода TextBox, в которое будет вводиться имя, можно назвать Name. Если имя должно отражать класс элемента управления, в имя можно включить дополнительную информацию. Тогда поле ввода TextBox, в которое вводится имя, будет называться NameTextBox или TextBoxName. Имя NameTextBox проще читать, но вариант TextBoxName позволяет сортировать по классу элементы управления в окне Properties (Свойства). Главное, придерживаться одного соглашения по именованию.

С точки зрения авторов этой книги возможность сортировки элементов управления по классу в окне Properties (Свойства) более важна, поэтому поле ввода называется TextBoxName. Для связи данных с листа с элементом управления TextBoxName свойство ControlSource установлено в значение Sheet1!B2. В следующей таблице показаны изменения, которые были внесены в свойства каждого элемента управления, расположенного в диалоговом окне PersonalData.

Элемент управления	*Имя*	*Свойство ControlSource*
TextBox	TextBoxName	Sheet1!B2
TextBox	TextBoxAge	Sheet1!B3
OptionButton	OptionButtonMale	Sheet1!C4
OptionButton	OptionButtonFemale	Sheet1!D4
CheckBox	CheckBoxMarried	Sheet1!B5
ListBox	ListBoxDepartment	
CommandButton	CommandButtonOK	

Назначив свойству ControlSource адрес ячейки на листе, можно связать элемент управления и ячейку. В результате формируется симметричная связь. Любое изменение ячейки оказывает влияние на элемент управления и любое изменение элемента управления меняет содержимое ячейки.

Описательные названия слева от полей ввода TextBox и над списком ListBox создаются с помощью элементов управления Label. Свойству Caption элементов управления Label присвоены значения Имя, Возраст и Подразделение. Свойству Caption фрейма вокруг переключателей присвоено значение Пол, а свойства Caption переключателей внутри фрейма имеют значения Мужской и Женский. Свойство Caption флажка установлено в значение Женат/Замужем.

Переключатели внутри фрейма не могут быть связаны с ячейкой B4. Непосредственное отображение значения переключателя не имеет смысла, поэтому функция `IF` в ячейке B4 выполняет преобразование значения `True` или `False` из ячейки C4 в строку “Мужчина” или “Женщина”.

```
=IF(C4=True, "Мужчина", "Женщина")
```

Хотя для получения необходимого результата достаточно установить значение в ячейке C4, для правильного отображения переключателей при выводе диалогового окна их необходимо связать с разными ячейками.

Свойство `RowSource` элемента управления `ListBoxDepartment` установлено в значение `Sheet1!A11:A18`. Вместо абсолютной адресации, как показано здесь, желательно назначать имена связанным ячейкам и использовать эти имена в свойствах `ControlSource`. Но в данном случае дополнительный этап создания имен ячеек был пропущен для упрощения примера.

Следующая процедура обработки события `Click` связана с кнопкой и находится в модуле кода диалогового окна `UserForm`:

```
Private Sub CommandButtonOK_Click()
    Call Unload(Me)
End Sub
```

`Me` является указателем на объект `UserForm`, в котором хранится этот код. Указатель `Me` может использоваться в любом модуле класса для ссылки на объект данного класса. Если элементы управления должны быть доступны из кода VBA позднее, то для сокрытия диалогового окна, не выгружая его из памяти, необходимо использовать метод `Hide`. Если применить метод `Unload`, диалоговое окно выгружается из памяти и значения элементов управления становятся недоступны. Примеры использования метода `Hide` показаны ниже.

Щелчок на кнопке [x] в верхнем правом углу диалогового окна `UserForm` также позволяет закрыть диалоговое окно. При этом оно выгружается из памяти.

Непосредственный доступ к элементам управления диалогового окна

> Связывание элементов управления диалогового окна с ячейками не всегда является лучшим решением. Большая гибкость достигается через непосредственный доступ к элементам управления диалогового окна UserForm. На рис. 9.3 показана модифицированная версия предыдущего примера. На экран выводится такое же диалоговое окно, но данные сохраняются в другой форме. Пол хранится в виде однобуквенного кода (M или F). Название подразделения (Department) хранится в виде двухбуквенного кода, как показано на рис. 9.3.

В диалоговое окно была добавлена кнопка **Отмена** (Cancel). Щелчок на ней позволяет отменить все изменения, которые были внесены пользователем. При этом автоматическое сохранение значений на листе не выполняется. Теперь в модуле кода диалогового окна `PersonalData` находится следующий код.

```
Option Explicit

Public Cancelled As Boolean
```

```
Private Sub CommandButtonCancel_Click()
  Cancelled = True
  Me.Hide
End Sub
Private Sub CommandButtonOK_Click()
  Cancelled = False
  Me.Hide
End Sub
```

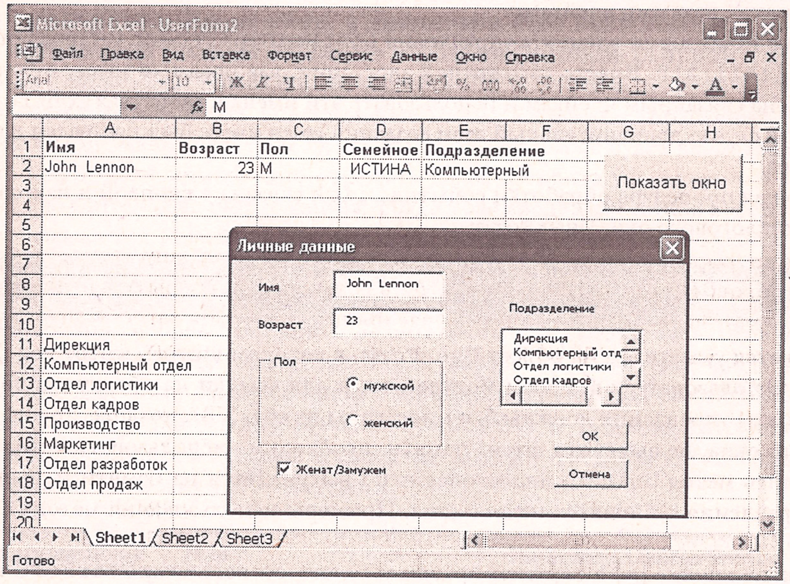


Рис. 9.3. Модифицированная версия диалогового окна UserForm

Открытая переменная `Cancelled` используется для определения щелчка на кнопке Отмена (Cancel). При щелчке на кнопке OK переменной `Cancelled` присваивается значение `False`. При щелчке на кнопке Отмена (Cancel) переменной `Cancelled` присваивается значение `True`. Щелчок на любой из кнопок приводит к сокрытию диалогового окна `PersonalData` без выгрузки из памяти. Следующая процедура также была добавлена в модуль кода диалогового окна `PersonalData`.

```
Private Sub UserForm_Initialize()

  Dim Departments As Variant
  Departments = VBA.Array("Дирекция", _
    "Компьютерный отдел", _
    "Отдел логистики", _
    "Отдел кадров", _
    "Производство", _
    "Маркетинг", _
    "Отдел разработок", _
    "Отдел продаж")

  Dim DepartmentCodes As Variant
  DepartmentCodes = VBA.Array("AD", _
    "CR", _
```

```
    "DS", _
    "HR", _
    "MF", _
    "MK", _
    "RD", _
    "SL")

  Dim Data(8, 2) As String
  Dim I As Integer
  For I = 0 To 7
    Data(I, 0) = Departments(I)
  Next I
  For I = 0 To 7
    Data(I, 1) = DepartmentCodes(I)

  Next

  ListBoxDepartment.List = Data

End Sub
```

Процедура обработки события `UserForm_Initialize` запускается при загрузке диалогового окна в память. Это событие не возникает при сокрытии и повторном отображении диалогового окна. В данном случае процедура обработки события используется для наполнения списка Подразделение двумя столбцами данных. Первый столбец содержит имя отдела, а второй — двухбуквенный код отдела.

`Departments` и `DepartmentCodes` являются массивами и наполняются значениями стандартным способом. Для этого используется функция `Array`. Функция `VBA.Array` позволяет создать массив с индексацией начиная с 0. Массив `Data` является динамическим массивом, а оператор `Dim` используется для установки размера массива `Data` в соответствии с размером массивов `Departments` и `DepartmentCodes`.

В цикле `For... Next` коды и названия отделов заносятся в массив `Data`, который используется для инициализации списка. При желании список названий и кодов отделов можно хранить на листе и устанавливать свойство `RowSource` списка равным диапазону листа. Такое решение было показано в предыдущем примере в этой главе.

При использовании списка `ListBox` с несколькими столбцами необходимо указать, какой столбец будет отображаться в связанной ячейке и возвращаться в качестве значения свойства `Value`. Это так называемый связанный столбец. Свойство `BoundColumn` списка `ListBox` (Подразделение) установлено в значение 1. Возможные значения свойства начинаются с 1, поэтому связанным считается столбец с кодами отделов. Так как в списке существует два столбца с данными, свойство `ColumnCount` установлено в значение 2.

Из-за произвольной ширины списка в примере виден только один столбец. Для указания ширины каждого столбца списка можно воспользоваться списком значений, разделенных точкой с запятой. Например, для сокрытия первого столбца и установки ширины второго столбца в 80 пикселей свойство `ColumnWidth` необходимо установить в значение 0;80. В данной реализации отображается полное имя и скрываются коды отделов, поэтому свойство `ColumnWidth` установлено в значение 93;0.

Следующий код находится в модуле кода листа `Sheet1`:

```
Option Explicit

Private Sub CommandButton1_Click()
Dim RangeData As Range
Dim Data As Variant
```

```
6:
7:   Set RangeData = Range("Database").Rows(2)
8:   Data = RangeData.Value
9:
10:  PersonalData.TextBoxName = Data(1, 1)
11:  PersonalData.TextBoxAge = Data(1, 2)
12:
14:    Case "F"
15:      PersonalData.OptionButtonFemale.Value = True
16:    Case "M"
17:      PersonalData.OptionButtonMale = True
18:   End Select
19:
20:   PersonalData.CheckBoxMarried.Value = Data(1, 4)
21:
22:   PersonalData.Show
23:    If (Not PersonalData.Cancelled) Then
24:      Data(1, 1) = PersonalData.TextBoxName
25:      Data(1, 2) = PersonalData.TextBoxAge
26:
27:      Select Case True
28:        Case PersonalData.OptionButtonFemale.Value
29:          Data(1, 3) = "F"
30:        Case PersonalData.OptionButtonMale.Value
31:          Data(1, 3) = "M"
32:      End Select
33:
34:      Data(1, 4) = PersonalData.CheckBoxMarried.Value
35:      Data(1, 5) = PersonalData.ListBoxDepartment.Text
36:      RangeData.Value = Data
37:    End If
38:
39:    Call Unload(PersonalData)
40: End Sub
```

Номера строк добавлены из-за большой длины листинга. При вводе этого кода в редакторе VBE номера строк вводить не нужно.

В строке 4 объявляется диапазон, а в строке 5 — переменная типа `Variant`. В строке 7 считывается диапазон `Database` и выделяется строка 2 этого диапазона. Значение диапазона присваивается переменной `Data` типа `Variant`. Фактически, значения с листа `Sheet1` были скопированы в локальный массив. В строках 10 и 11 выполняется копирование имени и возраста в соответствующие элементы управления диалогового окна. Оператор `Select Case` в строках с 13 по 18 проверяет, принадлежат ли данные мужчине или женщине, и устанавливает соответствующий переключатель на диалоговом окне. Достаточно изменить значение одного переключателя, так как использование фрейма позволяет рассматривать переключатели как группу. При этом в группе может быть установлен только один переключатель. В строке 20 флажок устанавливается в соответствии с семейным статусом.

К моменту выполнения оператора из строки 22 все данные уже были скопированы в диалоговое окно `PersonalData`, и оператор в строке 22 выводит диалоговое окно на экран. Если пользователь щелкнет на кнопке OK, условие в строке 23 запускает копирование значений с диалогового окна `PersonalData` в поля листа. Наконец, диалоговое окно выгружается из памяти. После разделения листа и диалогового окна пользователь получает возможность отмены внесенных изменений.

Отключение кнопки Закрыть

Одной из проблем показанного выше кода является возможность щелчка на кнопке Закрыть (Close) (кнопка [x] в верхнем правом углу диалогового окна). При этом процедура обработки событий не завершает работу и копирует изменения на лист. Это связано с тем, что по умолчанию переменная Cancelled имеет значение False. Обычно щелчок на кнопке [x] приводит к выгрузке формы, а значит, неудачным попыткам кода получить доступ к элементам управления диалогового окна.

В следующем примере используется процедура обработки события QueryClose. Она позволяет защитить диалоговое окно от закрытия, когда пользователь щелкает на кнопке [x]. Процедуру обработки события QueryClose можно применить для определения источника команды на закрытие и для отмены данного события при необходимости. Добавление следующего кода в модуль кода диалогового окна PersonalData позволяет отключить кнопку Закрыть (Close).

```
Private Sub UserForm_QueryClose(Cancel As Integer, CloseMode As Integer)
  If CloseMode = vbFormControlMenu Then
          Cancel = True  Beep
  End If
End Sub
```

Событие QueryClose возникает в четырех случаях. Причину его возникновения можно определить через сравнение параметра CloseMode со следующими встроенными константами:

Константа	*Значение*	*Причина возникновения события*
VbFormControlMenu	0	Пользователь щелкнул на кнопке [x] в меню управления
VbFormCode	1	Для выгрузки диалогового окна из памяти использован вызов Unload
VbAppWindows	2	Завершение операционной системы Windows
VbAppTaskManager	3	Приложение закрывается средствами Диспетчера задач Windows (Windows Task Manager)

Поддержка списка данных

Разработанный код можно расширить без дополнительных усилий для поддержки списка данных. Но в последнем примере использован другой подход. В этот раз весь код встроен в диалоговое окно PersonalData. В модуле кода листа хранится только код командной кнопки, щелчок на которой приводит к отображению диалогового окна. Код кнопки будет выглядеть следующим образом:

```
Private Sub CommandButton1_Click()
    PersonalData.Show
End Sub
```

Намного проще хранить данные в обычной СУБД, например Microsoft Access. Но если перед приложением не стоят особые требования, данные можно хранить на листе.

Если необходимо управлять более чем одной строкой данных, потребуется механизм добавления новых строк, удаления существующих и перемещения по строкам. При этом в диалоговое окно PersonalData придется добавить дополнительные элементы управления, показанные на рис. 9.4.

Были добавлены четыре кнопки, а список заменен на комбинированный. Кнопка Предыдущая запись используется для перемещения к предыдущей строке на листе. Кнопка Следующая запись применяется для перемещения к следующей строке на листе. Кнопка Новая запись необходима для добавления строк на лист, а кнопка Удалить запись — для удаления строк с листа. Элемент управления ComboBox используется для демонстрации альтернативного способа выбора элементов списка.

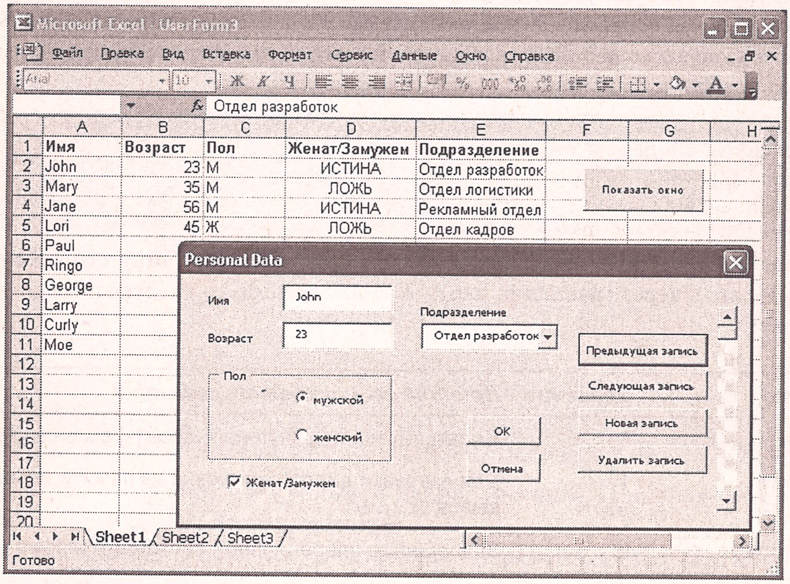


Рис. 9.4. Дополнительные элементы управления в диалоговом окне

Ниже рассматривается код диалогового окна PersonalData. Важно обратить внимание, что в начале модуля этого кода объявлены следующие переменные уровня модуля:

```
Dim aRangeData As Range
Dim Data As Variant
```

Эти переменные используются так же, как и в предыдущем примере. При этом можно менять ссылку на строку. Объектная переменная RangeData всегда указывает на текущую строку с данными в диапазоне Database. Имя Database присвоено диапазону A1:E11. Переменная Data всегда хранит значения диапазона RangeData в виде массива VBA.

Код процедуры обработки событий командной кнопки из предыдущего примера был преобразован в две вспомогательные процедуры, которые хранятся в модуле кода диалогового окна PersonalData.

```
Private Sub LoadRecord()
   ' Скопировать значения из RangeData на листе в массив vaData
   Data = RangeData.Value
   ' Присвоить массив значений элементам управления Personal
   TextBoxName.Value = Data(1, 1)
```

```
    TextBoxAge.Value = Data(1, 2)

    Select Case Data(1, 3)
        Case "F"
            OptionButtonFemale.Value = True
        Case "M"
            OptionButtonMale.Value = True
    End Select

    CheckBoxMarried.Value = Data(1, 4)

    ComboBoxDepartment.Value = Data(1, 5)

End Sub

Private Sub SaveRecord()
    ' Скопировать значения из элементов управления Personal
    ' в массив Data
    Data(1, 1) = TextBoxName.Value
    Data(1, 2) = TextBoxAge.Value

    Select Case True
        Case OptionButtonFemale.Value
            Data(1, 3) = "F"
        Case OptionButtonMale.Value
            Data(1, 3) = "M"
    End Select

    Data(1, 4) = CheckBoxMarried.Value
    Data(1, 5) = ComboBoxDepartment.Value

    ' Присвоить значения из массива Data текущей записи
    ' в диапазоне Database
    RangeData.Value = Data
End Sub
```

Так как этот код хранится в модуле кода диалогового окна PersonalData, ссылки на PersonalData при обращении к элементам управления необязательны.

Только процедуры LoadRecord и SaveRecord связаны со структурой данных и с элементами управления. Пока список данных называется Database, ни один фрагмент кода диалогового окна PersonalData не потребует внесения изменений при добавлении или удалении полей в набор данных. Кроме этого, данный код можно использовать вместе с другим набором данных. При этом достаточно будет перепроектировать элементы управления диалогового окна и модифицировать подпрограммы LoadRecord и SaveRecord.

Главным элементом навигации в окне PersonalData является полоса прокрутки, которая называется Navigator. Она используется другими кнопками для смены записи. Кроме того, полоса прокрутки доступна пользователю непосредственно. Свойство Value элемента управления Navigator соответствует номеру строки в диапазоне Database.

```
Private Sub Navigator_Change()
    ' При изменении значения полосы прокрутки сохранить
    ' текущую запись и загрузить
    ' запись, номер которой соответствует текущему
    ' значению полосы прокрутки
    Call SaveRecord
    Set RangeData = Range("Database").Rows(Navigator.Value)
    Call LoadRecord
End Sub
```

Если пользователь меняет значение свойства Navigator.Value (или значение свойства меняется в результате работы процедуры обработки события), то возникает событие Change, и текущая запись из диалогового окна PersonalData сохраняется на листе. При этом переменная RangeData переопределяется и указывает на строку диапазона Database, которая соответствует новому значению свойства Navigator.Value. После этого новая строка загружается в диалоговое окно PersonalData.

В этот раз в процедуру обработки события UserForm_Initialize внесены изменения. В данном случае процедура устанавливает правильное начальное значение переменной sbNavigator:

```
Private Sub UserForm_Initialize()
   'Устанавливает список значений Department
   'и загружает первую запись из диапазона Database
   Dim DepartmentCode As Variant
   Dim DepartmentList() As String
   Dim I As Integer

   DepartmentCode = VBA.Array("AD", _
         "CR", _
         "DS", _
         "HR", _
         "MF", _
         "MK", _
         "RD", _
         "SL", _
         "NA")

   ReDim DepartmentList(0 To UBound(DepartmentCode))

   For I = 0 To UBound(DepartmentCode)
      DepartmentList(I) = DepartmentCode(I)
   Next I

   ComboBoxDepartment.List = DepartmentList

   ' Загрузить первую запись из диапазона Database
   ' и инициализировать полосу прокрутки
   With Range("Database")
      Set RangeData = .Rows(2)
      Call LoadRecord
      Navigator.Value = 2
      Navigator.Max = .Rows.Count
   End With
End Sub
```

После инициализации свойства ComboBoxDepartment.List код инициализирует переменную RangeData, чтобы она указывала на вторую строку в диапазоне Database. Вторая строка диапазона содержит первую строку данных сразу после названий полей, которые расположены в первой строке. Данные из второй строки диапазона загружаются в диалоговое окно PersonalData. После этого свойство Value объекта Navigator устанавливается равным двум, а свойство Max объекта Navigator — равным количеству строк в диапазоне Database. Перемещение бегунка полосы прокрутки позволяет получать доступ ко всем строкам диапазона Database, начиная со второй строки.

Процедура обработки события Click для кнопки Следующая запись (Next Record) выглядит следующим образом:

```
Private Sub CommandButton2_Click()
    With Range("Database")
      If RangeData.Row < .Rows(.Rows.Count).Row Then
         'Загрузить следующую запись, если это не последняя запись
         Navigator.Value = Navigator.Value + 1
         'Обратите внимание: установка свойства Navigator.Value приводит
         'к запуску процедуры обработки события Change
      End If
   End With
End Sub
```

Условный оператор If сравнивает текущий номер строки в диапазоне Database с номером последней строки диапазона. Эта проверка позволяет защитить пользователя от перехода за пределы диапазона. Если перемещение еще возможно, значение объекта Navigator увеличивается на единицу. Это изменение приводит к появлению события Change для объекта Navigator. При этом в соответствующей строке диапазона сохраняются текущие данные из диалогового окна, сбрасывается значение переменной RangeData и в диалоговое окно загружаются данные из следующей строки.

Код обработки события для кнопки Предыдущая запись (Previous Record) выглядит так же, как и код кнопки Следующая запись (Next Record). (Здесь для разнообразия показан код процедуры без использования оператора With.)

```
Private Sub CommandButton1_Click()
  If RangeData.Row > Range("Database").Rows(2).Row Then
    'Загрузить предыдущую запись, если это не первая запись.
    Navigator.Value = Navigator.Value - 1
    ' Обратите внимание: установка свойства Navigator.Value
    ' приводит к запуску процедуры обработки события Change
  End If
End Sub
```

Эта проверка позволяет защитить пользователя от перехода выше второй строки диапазона Database. Условный оператор If, сравнивающий номер текущей строки с границами диапазона, мог быть реализован с помощью свойств Value, Max и Min полосы прокрутки Navigator, но здесь реализован метод определения номера последней строки именованного диапазона. Иногда этот прием оказывается очень полезной. Проверки в данном случае обязательны. Если установить значение свойства Navigator.Value больше значения свойства Max или меньше значения свойства Min диапазона, приложение выдаст сообщение об ошибке времени выполнения.

Ниже показан код обработки события для кнопки Удалить запись (Delete Record).

```
Private Sub CommandButton4_Click()
   ' Удаляет текущую запись в PersonalData
   If Range("Database").Rows.Count = 2 Then
      ' Не удалять, если осталась только одна запись
      MsgBox "Удаление всех записей невозможно", vbCritical
      Exit Sub
   ElseIf RangeData.Row = Range("Database").Rows(2).Row Then
      ' Если текущей является первая запись, переместиться
      ' на одну запись ниже и удалить первую запись
      'сместив нижние строки для заполнения пустоты
      Set RangeData = RangeData.Offset(1)
      RangeData.Offset(-1).Delete shift:=xlUp
      Call LoadRecord
   Else
      ' Если запись не первая, перед удалением перейти на следующую запись
      Navigator.Value = Navigator.Value - 1
```

```
      'Обратите внимание: установка значения sbNavigator.Value приводит
      'к запуску процедуры обработки события Change
      RangeData.Offset(1).Delete shift:=xlUp
   End If
   Navigator.Max = Navigator.Max - 1
End Sub
```

На эту процедуру возложены следующие задачи:

- процедура завершает выполнение при попытке удаления последней записи в диапазоне Database;
- при попытке удаления первой записи полю RangeData присваивается ссылка на вторую запись. Значение свойства Navigator.Value не сбрасывается, так как после удаления строки 1 строка 2 превращается в строку 1. Процедура LoadRecord вызывается для загрузки данных из строки RangeData в диалоговое окно UserForm;
- при удалении не первой записи значение свойства Navigator.Value уменьшается на единицу. При этом в диалоговое окно UserForm загружается следующая запись;
- в завершение процедуры счетчик количества строк в диапазоне Database, который хранится в свойстве Navigator.Max, уменьшается на единицу.

Ниже показан код процедуры обработки события для кнопки Новая запись (New Record).

```
Private Sub CommandButton3_Click()
  ' Добавить новую запись в конец базы данных
  Dim RowCount As Integer

  With Range("Database")
    ' Добавить строку в базу данных
    RowCount = .Rows.Count + 1
    .Resize(RowCount).Name = "Database"
    Navigator.Max = iRowCount
    Navigator.Value = iRowCount
    ' Обратите внимание: установка значения sbNavigator.Value
    ' приводит к запуску процедуры обработки события Change
  End With

  ' Установка значения по умолчанию
  OptionButtonMale.Value = True

  CheckBoxMarried = False
  CheckBoxDepartment.Value = "NA"
End Sub
```

В этой процедуре обработки события переменной RowCount присваивается значение, на единицу превышающее количество строк в диапазоне Database. После этого генерируется ссылка на диапазон с большим количеством строк, чем в диапазоне Database, и имя Database присваивается новому диапазону. После этого значение переменной RowCount присваивается свойству Value и свойству Max объекта Navigator. При этом новая пустая строка становится текущей и в диалоговое окно PersonalData загружаются пустые значения. При этом некоторые элементы управления диалогового окна получают принятые по умолчанию значения.

Осталось рассмотреть код обработки событий для кнопок OK и Отмена (Cancel).

```
Private Sub CommandButtonCancel_Click()
    Cancelled = True
    Me.Hide
```

```
End Sub

Private Sub CommandButtonOK_Click()
    Cancelled = False
    SaveRecord
    Me.Hide
End Sub
```

Обе процедуры выгружают диалоговое окно PersonalData из памяти. При этом щелчок на кнопке OK приводит к сохранению изменений из диалогового окна UserForm в текущую строку диапазона.

Немодальные диалоговые окна UserForm

В Excel 2000, Excel 2002 и Excel 2003 предоставляется возможность вывода немодальных диалоговых окон UserForm. Модальные диалоговые окна UserForm, которые рассматривались до этого, не поддерживали переключение фокуса, пока диалоговое окно отображается на экране. При этом, если не скрыть или не выгрузить диалоговое окно из памяти, нельзя было активизировать листы, меню или панели инструментов. Если в процедуре для вывода диалогового окна используется вызов метода Show, пока диалоговое окно не будет скрыто или выгружено из памяти, операторы после вызова метода Show выполняться не будут.

Немодальные диалоговые окна UserForm позволяют активизировать листы, меню и панели инструментов. Немодальное диалоговое окно остается на переднем плане, пока не будет скрыто или выгружено из памяти. Если в процедуре для вывода диалогового окна используется метод Show, операторы после вызова метода Show будут выполняться немедленно после вызова метода. Диалоговое окно PersonalData из предыдущего примера также можно вывести в качестве немодального диалогового окна. Для этого достаточно внести следующее изменение в код:

```
Private Sub CommandButton1_Click()
    Call PersonalData(vbModeless)
End Sub
```

При отображении немодального диалогового окна UserForm можно выполнять другие задачи. Поддерживается даже копирование данных между полями ввода в диалоговом окне и ячейками листа.

Важно отметить, что в последнем примере лист не связывается с диалоговым окном. Данные копируются из листа по требованию. То есть, если вывести немодальное диалоговое окно PersonalData и изменить данные на листе, то это изменение не будет автоматически отражено в диалоговом окне. В данном случае потребуется создание механизма уведомления диалогового окна UserForm о модификации данных на листе. Сможете придумать способ обновления диалогового окна UserForm в случае изменения данных на листе? (Вот подсказка: обратите внимание на событие Change объекта Worksheet.)

Резюме

В этой главе рассматривалось, как создавать и использовать модальные и немодальные диалоговые окна. На примерах было продемонстрировано, как ссылаться на данные в диапазонах листа или копировать данные из диапазонов. Несколько обработчиков со-

бытий применялись для добавления и удаления данных с листа электронной таблицы, что позволило использовать лист в качестве базы данных.

В любом случае, электронная таблица хорошо подходит для управления данными и числами в ячейках, но намного слабее справляется с обязанностями базы данных. Сложность заключается в том, что технологии баз данных значительно развились за последние десять лет и теперь поддерживают проверку действительности, ограничения, сложные отношения, индексацию, возможности поиска и другие возможности. Если от приложения действительно требуется управление данными, стоит обратить внимание на базу данных Microsoft Access. Существуют и другие поставщики баз данных, но язык VBA, который изучается в контексте Excel, работает и в Access.

Кроме этого, Excel и Access могут взаимодействовать друг с другом средствами языка VBA. В связи с этим рекомендуется использовать Access для управления нечисловыми данными и Excel — для обработки чисел. При необходимости можно создать диалоговые окна и управлять именами и числами в Excel, но база данных Access намного лучше справляется с такими задачами. Однако в обработке чисел никому не превзойти Excel.

Красота Microsoft Office заключается в мощности и завершенности каждого отдельного компонента. Кроме этого, любой компонент может общаться с другими компонентами с помощью общего языка, VBA. Если воспринимать Excel, как мощный механизм обработки числовых данных и большой компонент, то его можно использовать для создания изолированных решений с применением диалоговых окон или в качестве мощного математического механизма для интегрированных решений уровня предприятия.

10

Добавление элементов управления

Как было показано в главе 1, на листы Excel добавляются элементы управления двух типов. Можно использовать элементы управления ActiveX, которые доступны на панели инструментов **Элементы управления** (Control Toolbox), или же элементы управления, доступные на панели инструментов **Формы** (Forms). Панель инструментов **Формы** (Forms) была представлена в Excel 5 и Excel 95. Предоставленные на этой панели элементы управления предназначены для диалоговых листов, используемых в этих версиях Excel. Кроме того, такие элементы управления могут быть интегрированы в лист или диаграмму. После выхода Excel 97 вместо диалоговых листов можно применять диалоговые окна `UserForm`. В диалоговых окнах `UserForm` используются элементы управления ActiveX.

Панели инструментов

В Excel все еще поддерживаются диалоговые листы и элементы управления на панели инструментов **Формы** (Forms) (рис. 10.1). При этом элементы управления с панели **Формы** (Forms) обладают некоторыми преимуществами по сравнению с элементами управления ActiveX.

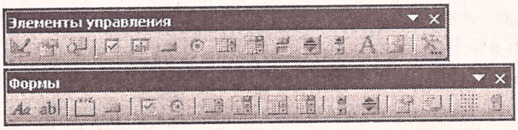


Рис. 10.1. Панели инструментов Формы и Элементы управления

Элементы управления с панели **Формы** (Forms) намного проще элементов управления ActiveX. Кроме этого, в диаграммы можно встраивать только элементы управления с панели **Формы** (Forms). Каждый из этих элементов реагирует только на одно событие. В большинстве случаев это событие `Click`. Исключением является поле ввода, которое реагирует на событие `Change`.

Если элемент управления и процедуры обработки событий необходимо определить не вручную, а в коде VBA, проще использовать элементы управления с панели **Формы** (Forms). Они обладают значительным преимуществом перед элементами управления ActiveX, а именно: предоставляют возможность размещения процедуры обработки события в стандартном модуле, поддерживают использование любого действительного имени процедуры VBA и обеспечивают возможность создания процедуры в процессе написания кода приложения еще до создания элементов управления.

Элемент управления можно создавать программно, как только в нем возникает необходимость. При этом имя процедуры обработки события присваивается свойству `OnAction` элемента управления. Одну и ту же процедуру обработки события можно назначать нескольким элементам управления. С другой стороны, процедуры обработки событий элементов управления ActiveX должны храниться в модуле класса листа или диалогового окна `UserForm`, в которые они встроены. В этом случае имя процедуры обработки события должно состоять из имени элемента управления и имени события. Например, процедура обработки события `Click` элемента управления `OptionButton1` определяется так:

```
Sub OptionButton1_Click()
```

При попытке создания процедуры обработки события до создания элемента управления ActiveX и попытке сослаться на элемент управления в коде процедуры выдаются сообщения об ошибках компиляции. В результате процедуры обработки событий придется создавать программно, а это далеко не тривиальная задача, как будет показано дальше. Кроме этого, в главе 14 приведен пример программного добавления процедуры обработки события к элементу управления в диалоговом окне `UserForm`.

С другой стороны, процедура обработки события для элемента управления с панели **Формы** (Forms) может иметь любое имя и для определения ссылки на вызвавший элемент управления может использовать свойство `Caller` объекта `Application`. Как будет показано далее в этой главе, имя элемента управления не обязательно должно присутствовать в имени процедуры обработки события или в ссылках на элемент управления.

Элементы управления ActiveX

На рис. 10.2 показано четыре типа встроенных в лист элементов управления ActiveX (книга `Controls.xls` доступна для скачивания на сайте `http://www.wrox.com`). В данном случае реализованы следующие элементы управления: полоса прокрутки в ячейках C3:F3 устанавливает значение в ячейке B3, кнопки счетчика в ячейке C4 позволяют увеличить процент роста в ячейке B4, установка флажка в ячейке B5 позволяет увеличить ставку налога в ячейке B16 с 30% до 33%, а переключатель в столбце I позволяет изменить стоимость в ячейке B15. Кроме этого, переключатель дает возможность изменить минимальное и максимальное значение полосы прокрутки.

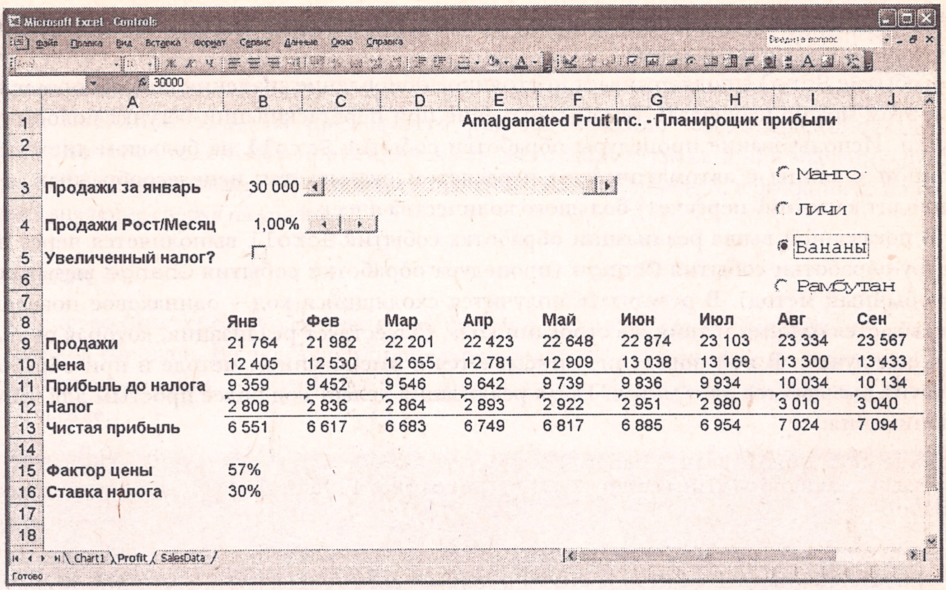


Рис. 10.2. Использование элементов управления ActiveX

Элемент управления ActiveX может быть связан с ячейкой на листе. Для этого используется свойство `LinkedCell`. В результате в ячейке всегда будет отображаться значение свойства `Value` объекта элемента управления. Ни один из элементов управления на рис. 10.2 не связан с ячейкой на листе, хотя полосу прокрутки можно было бы связать с ячейкой, так как значение этой полосы выводится в ячейке B3. Для обновления содержимого ячеек в каждом элементе управления применяется процедура обработки события. Такой подход позволяет добиться большей гибкости, чем при использовании простой связи с ячейкой. При этом экономится ячейка листа.

Полоса прокрутки

Для вывода значения свойства `Value` в ячейке B3 полоса прокрутки использует процедуры обработки событий `Change` и `Scroll`. Максимальное и минимальное значения полосы прокрутки устанавливаются с помощью переключателей (которые будут рассматриваться ниже):

```
Private Sub ScrollBar1_Change()
   Range("B3").Value = ScrollBar1.Value
End Sub

Private Sub ScrollBar1_Scroll()
   ScrollBar1_Change
End Sub
```

Процедура обработки события `Change` выполняется при изменении значения полосы прокрутки. Для этого нужно щелкнуть выше или ниже бегунка полосы прокрутки (если используется горизонтальная полоса прокрутки, можно щелкать слева или справа) или перетащить его. Но сразу после использования переключателя возникает небольшая ошибка. Перетаскивание бегунка не приводит к появлению события `Change` при первой

попытке. Лишь использование процедуры обработки события Scroll приведет поведение полосы прокрутки в норму.

Событие Scroll поддерживает постоянное обновление значения полосы прокрутки. При этом можно наблюдать текущее значение при перетаскивании бегунка полосы прокрутки. Использование процедуры обработки события Scroll на большом листе электронной таблицы с автоматическим пересчетом оказывается нецелесообразным, если приводит к частому пересчету большого количества ячеек.

В показанной выше реализации обработка события Scroll выполняется через процедуру обработки события Change (процедура обработки события Change вызывалась как обычный метод). В результате получится сходящийся код — одинаковое поведение описывается одними и теми же строками кода. Существует реализация, которая оказывается еще лучше. В ней поведение описывается в именованном методе и при обработке события вызывается этот метод. Такая реализация делает код более простым для чтения и понимания:

```
Private Sub ScrollBar1_Change()
    Call SetRangeValue(Range("B3"), ScrollBar1.Value)
End Sub

Private Sub ScrollBar1_Scroll()
    Call SetRangeValue(Range("B3"), ScrollBar1.Value)
End Sub

Private Sub SetRangeValue(ByVal R As Range, ByVal Value As Double)
    R.Value = Value
End Sub
```

Такой код связан с определенными накладными расходами за счет опосредованного вызова подпрограммы SetRangeValue, но в результате код становится более ясным, а изменение в подпрограмме SetRangeValue отражено во всех фрагментах кода, где используется эта подпрограмма. Важно помнить, что причиной создания этого кода является попытка снижения стоимости владения, а не стоимости с точки зрения циклов центрального процессора.

Счетчик

Элемент управления SpinButton применяет события SpinDown и SpinUp для уменьшения или увеличения значения ячейки B4. Для реализации поведения этого элемента управления используется прием, который применялся в конце предыдущего раздела:

```
' Здесь используется подпрограмма SetRangeValue из предыдущего
' разделы
Private Property Get SpinRange()
    Set SpinRange = Range("B4")
End Property

Private Sub SpinButton1_SpinDown()
    Call SetRangeValue(SpinRange, WorksheetFunction.Max(0,
↳ SpinRange.Value - 0.0005))
End Sub

Private Sub SpinButton1_SpinUp()
    Call SetRangeValue(SpinRange, WorksheetFunction.Min(0.01,
↳ SpinRange.Value + 0.0005))
End Sub
```

Доступное только для чтения свойство SpinRange используется для того, чтобы заставить методы, работающие с элементом управления SpinButton, обращаться к одному и тому же диапазону. События SpinUp и SpinDown реализованы через повторное использование метода SetRangeValue из предыдущего раздела. Функции WorksheetFunction.Max и WorksheetFunction.Min применяются для увеличения и уменьшения значения диапазона на 0.0005. Функция Max ограничивает значение снизу на уровне 0, а функция Min ограничивает значение сверху на уровне 1.

Флажок

Свойство CheckBox.Value будет равно True, если флажок установлен, и False, если флажок сброшен. В следующей процедуре обработки события Click реализовано переключение значения ячейки B16 с 33% при установленом флажке на 30% при сброшенном флажке:

```
Private Sub CheckBox1_Click()
    If CheckBox1.Value Then
        Range("B16").Value = 0.33
    Else
        Range("B16").Value = 0.3
    End If
End Sub
```

Существует более интересная реализация. Опытные программисты на языке C могут создать следующий код. Однако создания такого кода стоит избегать, так как он может скрыть смысл алгоритма, но все же стоит научиться понимать данный код.

```
Private Sub CheckBox1_Click()
    ' Интересная реализация
    Range("B16").Value = Array (0.3, 0.33) (Abs(CheckBox1.Value))
```

Как создать самодокументированную версию первой процедуры обработки события Click? (Подсказка: создайте именованный метод, который описывает происходящее, то есть, переключает значение диапазона Range("B16").)

Переключатель

Событие Click является общим для многих элементов упраления. Элемент управления OptionButton использует событие Click точно так же, как элемент управления CheckBox. Это событие возникает в результате нажатия и отпускания левой кнопки мыши (или нажатия клавиши пробела, если фокус установлен на переключатель). В данном случае каждая процедура обработки события OptionButton.Click вызывает метод SetOption. Любая процедура обработки события OptionButton.Click реализована так же, как для элемента управления OptionButton1:

```
Private Sub OptionButton1_Click()
    Call SetOptions()
End Sub
```

Обработка всех переключателей выполняется в следующей процедуре. Эта процедура хранится в модуле класса листа Profit, в котором расположены процедуры обработки событий OptionButton.Click:

```
Private Sub SetOptions()

    Select Case True
```

```
        Case OptionButton1.Value
            Call SetCostFactor(0.63)
            Call SetScrollMinMax(50000, 150000)
        Case OptionButton2.Value
            Call SetCostFactor(0.74)
            Call SetScrollMinMax(25000, 75000)
        Case OptionButton3.Value
            Call SetCostFactor(0.57)
            Call SetScrollMinMax(10000, 30000)
        Case OptionButton4.Value
            Call SetCostFactor(0.65)
            Call SetScrollMinMax(15000, 30000)
    End Select

End Sub

Private Property Get CostFactorRange()
    Set CostFactorRange = Range("B15")
End Property

Private Sub SetCostFactor(ByVal CostFactor As Double)
    Call SetRangeValue(CostFactorRange, CostFactor)
End Sub

Private Sub SetScrollMinMax(ByVal Min As Long, ByVal Max As Long)
    ScrollBar1.Min = Min
    ScrollBar1.Max = Max
    ScrollBar1.Value = Max
End Sub
```

(Процедура SetRangeValue была позаимствована из предыдущих разделов.) Структура Select Case применяется не обычным образом. Обычно ссылка на переменную используется в первой строке Select Case, а значения для сравнения указываются в строках Case. В данном случае в строке Select Case используется значение True, а ссылка на свойство Value переключателей указываются в строках Case. Эта структура хорошо подходит для обработки набора переключателей, только один из которых может иметь значение True.

На показанном выше листе только один переключатель имеет значение True, так как все переключатели объединены в группу. При добавлении переключателей на лист значение свойства GroupName устанавливается равным имени листа — Profit. Если необходимо создать два набора независимых переключателей, переключателям из каждой группы придется назначить разное значение свойства GroupName.

Элементы управления с панели Формы (Forms)

На рис. 10.3 показан элемент управления панели инструментов **Формы** (Forms), который используется для выбора имени продукта в столбце D. Элемент управления появляется при двойном щелчке на любой ячейке в этом столбце. При выборе продукта его имя вводится в ячейку, которая находится "под" элементом управления. При этом цена продукта вводится в столбец F в той же строке, и элемент управления исчезает.

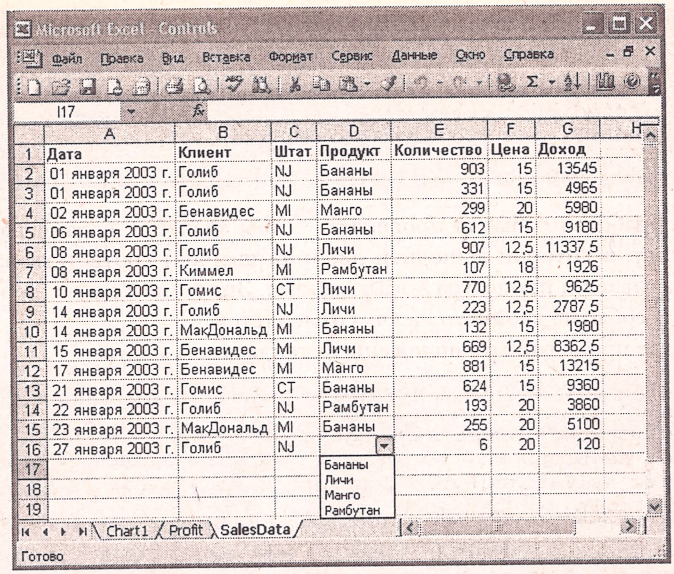


Рис. 10.3. Элемент управления с панели Формы

Если навести курсор на кнопку панели **Формы** (Forms), позволяющую создать элемент управления, в появившейся экранной подсказке элемент управления будет описан как комбинированный список (`ComboBox`). В объектной модели Excel этот элемент управления называется раскрывающимся списком (`DropDown`).

> Объект `DropDown` является скрытым членом объектной модели Excel 97 и более поздних версий. В справочном руководстве отсутствуют статьи об этом элементе управления и информация о нем не выводится в окне Object Browser (Просмотр объектов). Для отображения информации об объекте `DropDown` в окне Object Browser (Просмотр объектов) щелкните правой кнопкой мыши в окне Object Browser (Просмотр объектов) и выберите пункт Show Hidden Members (Отобразить скрытые компоненты) из контекстного меню. Значительный объем информации об элементах управления Формы (Forms) можно получить через запись макросов и использование окна Object Browser (Просмотр объектов), однако для получения полной документации необходим доступ к Excel 5 или Excel 95.

Элемент управления раскрывающегося списка создается процедурой, которая вызывается из процедуры обработки события `BeforeDoubleClick` на листе `SheetData`. Лист имеет программное имя `Sheet2`.

```
Option Explicit

Private Sub Worksheet_BeforeDoubleClick(ByVal Target As Range, _
    Cancel As Boolean)

    If (IsColumnSelected(Target, "D")) Then
        Call AddDropDown(Target)
        Cancel = True
    End If

End Sub
```

```
Private Function IsColumnSelected(ByVal Target As Range, _
    ByVal ColumnName As String) As Boolean

    IsColumnSelected = _
        (Intersect(Target, Columns(ColumnName)) Is Nothing = False)

End Function
```

Процедура обработки события проверяет, чтобы значение параметра `Target` (описывающего ячейку, в которой выполнен двойной щелчок) соответствовало столбцу D. Если это так, вызывается процедура `AddDropDown`. Значение `Target` передается в качестве параметра, а событие двойного щелчка отменяется.

Следующие две процедуры расположены в стандартном модуле:

```
Option Explicit

Public Sub AddDropDown(Target As Range)
    Dim Control As DropDown
    Dim Products As Variant
    Dim I As Integer

    Products = Array("Бананы", "Лихнис", "Манго", "Рамбутан")
    Set Control = SalesData.DropDowns.Add( _
        Target.Left, Target.Top, Target.Width, Target.Height)

    Control.OnAction = "EnterProductInformation"

    For I = LBound(Products) To UBound(Products)
        Call Control.AddItem(Products(I))
    Next I
End Sub

Private Sub EnterProductInformation()
    Dim Prices As Variant
    Prices = Array(15, 12.5, 20, 18)

    With SalesData.DropDowns(Application.Caller)
        .TopLeftCell.Value = .List(.ListIndex)
        .TopLeftCell.Offset(0, 2).Value = _
Prices(.ListIndex + LBound(Prices) - 1)
        .Delete
    End With
End Sub
```

Процедура `AddDropDown` не объявлена как закрытая, так как ее нельзя будет вызывать из объекта `SalesData`. Это может стать проблемой, если пользователям нельзя просматривать процедуру в диалоговом окне Сервис⇨Макрос⇨Макросы (Tools⇨Macro⇨Macros), но так как этот метод принимает аргумент, в этом диалоговом окне он не отображается. Кроме того, подпрограмму `AddDropDown` можно разместить в модуле `SalesData` или в стандартном модуле. Подпрограмма будет работать в любом случае.

Для создания раскрывающегося списка в подпрограмме `AddDropDown` используется метод `Add` коллекции `DropDowns`. Элемент управления выравнивается в соответствии с элементом управления, на который указывает параметр `Target`. В результате выравнивания раскрывающийся список получает те же значения свойств `Left`, `Top`, `Width` и `Height`, что и соответствующая ячейка. В подпрограмме `AddDropDown` свойству `DropDown.OnAction` присваивается указатель на подпрограмму `EnterProductInformation`. Это значит, что под-

программа `EnterProductInformation` будет запускаться при выборе элемента управления из раскрывающегося списка. Для добавления элементов из диапазона `Products` в раскрывающийся список в цикле `For... Next` используется метод `AddItem`.

Подпрограмма `EnterProductInformation` объявлена с квалификатором доступа `Private`. Это защищает ее от доступа через диалоговое окно Сервис⇨Макрос⇨Макросы (Tools⇨Macro⇨Macros). Хотя подпрограмма является закрытой, она доступна для раскрывающегося списка. Подпрограмму `EnterProductInformation` можно разместить в модуле `SalesData`, но свойству `OnAction` раскрывающегося списка придется присваивать значение `SalesData.EnterProductInformation`.

Подпрограмма `EnterProductInformation` загружает цены соответствующих продуктов в массив `Prices`. После этого вызов `Application.Caller` используется для получения имени элемента управления раскрывающегося списка, который вызвал событие `OnAction`. Это имя применяется в качестве индекса коллекции `DropDowns` при получении ссылки на объект `DropDown` в модуле `SalesData`. Для получения индекса выбранного элемента раскрывающегося списка в конструкции `With... End With` в подпрограмме `EnterProdInformation` используется свойство `ListIndex`.

Непосредственный доступ к имени выбранного в раскрывающемся списке `DropDown` объекта невозможен. Это поведение отличается от поведения объекта `ComboBox`, который возвращает имя выбранного пункта в качестве значения свойства `Value`. Свойство `Value` раскрывающегося списка имеет то же значение, что и свойство `ListIndex`, и содержит номер позиции выбранного пункта списка. Для получения имени выбранного элемента раскрывающегося списка необходимо воспользоваться значением свойства `ListIndex` в качестве индекса свойства `List` (индексация свойства начинается с 1). Свойство `List` возвращает массив всех элементов списка.

Свойство `TopLeftCell` объекта `DropDown` возвращает ссылку на объект `Range`, который находится под верхним левым углом объекта `DropDown`. Подпрограмма `EnterProdInformation` присваивает выбранный элемент списка свойству `Value` данного объекта `Range`. После этого цена продукта сохраняется в объекте `Range`, расположенном на два столбца вправо от объекта `Range`, указанного в свойстве `TopLeftCell`.

Кроме этого в подпрограмме `EnterProdInformation` в качестве индекса для массива `Prices` используется свойство `ListIndex` раскрывающегося списка. В данном случае проблема заключается в индексации раскрывающегося списка начиная с 1, в то время как список функции `Array` зависит от оператора `Option Base` в начале модуля. Оператор `LBound(Prices)-1` используется для уменьшения значения `ListIndex` на единицу, если применяется оператор `Option Base 0`, и для уменьшения на 0, если используется оператор `Option Base 1`.

Следующий код применяется для обеспечения индексации начиная с 0 при использовании оператора `Option Base 1` в Excel 97 и более поздних версиях:

```
Prices = VBA.Array(15, 12.5, 20, 18)
```

Этот прием не работает в Excel 5 и Excel 95, так как приведенный код зависит от оператора `Option Base`.

Динамические элементы управления ActiveX

Как было показано ранее, элементы управления ActiveX сложнее в программировании, чем элементы управления с панели инструментов Формы (Forms). В то же время, элементы управления ActiveX предоставляют больше возможностей, а значит их изучение имеет определенный смысл. Ниже будет показано, как создать комбинированный список, который ведет себя как комбинированный список из предыдущего примера. Для того чтобы пример немного отличался, воспользуемся событием `BeforeRightClick` для включения комбинированного списка в столбце D на листе `SalesData`.

```
Private Const ControlName As String = "Combo"
Private Sub Worksheet_BeforeRightClick( _
    ByVal Target As Range, Cancel As Boolean)

    Dim Ole As OLEObject
    Dim Control As MSForms.ComboBox
    Dim Line As Long
    Dim CodeModule As Object

    If Not IsColumnSelected(ActiveCell, "D") Then Exit Sub

    ' Отключить обновление экрана при добавлении элемента управления
    Application.ScreenUpdating = False

    ' Определить необходимость создания раскрывающегося списка
    On Error Resume Next

        Set Ole = Me.OLEObjects(ControlName)
        If Ole Is Nothing = False Then
            GoTo Finish
        End If

    On Error GoTo 0

    ' Доставить раскрывающийся список в активную ячейку
    Set Ole = Me.OLEObjects.Add( _
        ClassType:="Forms.ComboBox.1", Link:=False, _
        DisplayAsIcon:=False, Left:=ActiveCell.Left,
↳Top:=ActiveCell.Top, _
        Width:=ActiveCell.Width, Height:=ActiveCell.Height)

    Ole.Name = ControlName
    Set Control = Ole.Object
    Control.Name = ControlName

    Call Control.AddItem("Бананы")
    Call Control.AddItem("Лихнис")
    Call Control.AddItem("Манго")
    Call Control.AddItem("Рамбутан")

    ' Создать процедуру обработки события для щелчка
    ' на комбинированном списке

Set CodeModule = _
        ThisWorkbook.VBProject.VBComponents(CodeName).CodeModule

        Line = CodeModule.CreateEventProc("Click", ControlName)
```

```
        Call CodeModule.ReplaceLine(Line + 1, " ProcessComboClick")

    ' Убедиться, что окно Excel активно
    Application.Visible = False
    Application.Visible = True

Finish:
    Cancel = True
    Application.ScreenUpdating = True

End Sub
```

Сначала необходимо убедиться, что событие возникло в столбце D. Кроме этого, нужно быть уверенным в отсутствии еще одного комбинированного списка на листе. Существование такого комбинированного списка означает, что пользователь создал список, но еще не выбрал один из элементов. Проверка такой ситуации не требовалась в предыдущем примере, так как комбинированные списки были независимы, хотя и использовали одну и ту же процедуру обработки события OnAction. Элементы управления ActiveX не могут одновременно применять процедуру обработки события Click, поэтому необходимо обеспечить уникальность комбинированного списка на листе.

Элемент управления ActiveX будет называться Combo. Самым быстрым способом определения существования элемента управления с именем Combo является создание ссылающейся на него объектной переменной. Если попытка завершается неудачно, элемент управления не существует. Для защиты макроса от остановки и выдачи сообщения об ошибке в случае отсутствия элемента управления используется код обработки ошибки. Перед завершением подпрограммы можно было бы вывести сообщение с описанием, но это не является основным назначением этого примера. Установка параметра Cancel в значение True подавляет появление контекстного меню при щелчке правой кнопкой мыши.

Если все проходит нормально, новый комбинированный список появляется в активной ячейке. Стоит обратить внимание, что объект ActiveX не добавляется непосредственно на лист. Объект хранится в объекте OLEObject точно так же, как встроенная диаграмма — в объекте ChartObject (дополнительная информация о встроенных диаграммах приводится в главе 24). Возвращаемое значение метода Add коллекции OLEObjects присваивается переменной Ole. Это сделано для того, чтобы упростить обращение к объекту OLEObject в дальнейшем. Для упрощения идентификации свойству Name объекта Ole присваивается строковое значение "Combo".

После этого создается объектная переменная Control, которая ссылается на объект ComboBox, содержащийся в объекте Ole. Объект OLEObject возвращается свойством Object объекта Ole. В следующей строке кода объекту ComboBox присваивается имя "Combo". В Excel 2000, Excel 2002 и Excel 2003 эта операция необязательна. При присвоении имени объекту OLEObject оно автоматически присваивается встроенному объекту. Такое поведение недоступно в Excel 97, поэтому имя должно присваиваться явно.

Далее создается процедура обработки события Click для комбинированного списка. Процедуру обработки события нельзя создать заранее. Если объект ActiveX, на который ссылается процедура, не существует, при компиляции будут выданы сообщения об ошибке. Методика программного создания процедур обработки событий подробно рассматривается в главе 14.

Переменной CodeModule присваивается ссылка на модуль класса листа, а метод CreateEventProc из модуля кода используется для ввода первой и последней строки процедуры обработки события Combo_Click с пустой строкой между ними. Метод воз-

вращает номер первой строки процедуры, который присваивается переменной Line. Метод ReplaceLine заменяет пустую вторую строку процедуры вызовом подпрограммы, называемой ProcessComboClick. Эта процедура показана ниже. Код процедуры ProcessComboClick уже существует в модуле кода листа.

К сожалению, при добавлении кода в модуль кода, как сделано в этом случае, выполняется активизация модуля кода и пользователь может остаться один на один перед экраном, полным исходного кода. Если свойство Visible приложения Excel сначала установить в значение False, а потом — в значение True, то при завершении процедуры окно Excel станет видимым. Хотя обновление экрана было отключено в начале процедуры, экран может мерцать. Существует возможность подавления такого мерцания. Для этого необходимо воспользоваться вызовами Windows API (дополнительная информация об использовании Windows API приводится в главе 16).

Процедура обработки события Click, созданная показанным ранее кодом, выглядит следующим образом:

```
Private Sub Combo_Click()
    ProcessComboClick
End Sub
```

При выборе элемента комбинированного списка вызывается процедура обработки события Click, которая, в свою очередь, вызывает процедуру ProcessComboClick. Процедура ProcessComboClick хранится в модуле кода листа и содержит следующий код:

```
Private Sub ProcessComboClick()
    Dim Line As Long
    Dim CodeModule As Object

    ' Ввести выбранное значение
    With OLEObjects(ControlName)
        .TopLeftCell.Value = .Object.Value
        .Delete
    End With

    ' Удалить процедуру обработки события Click
    ' для комбинированного списка
        Set CodeModule = _
            ThisWorkbook.VBProject.VBComponents(CodeName).CodeModule

        Line = CodeModule.ProcStartLine("Combo_Click", 0)
        Call CodeModule.DeleteLines(Line, 4)

End Sub
```

Комбинированный список хранится в виде объекта в объекте OLEObject, называемом Combo. В показанном выше коде выбранное значение из комбинированного списка копируется в ячейку под списком. После этого объект OLEObject и его содержимое удаляются.

Потом код удаляет процедуру обработки события. Переменной CodeModule присваивается ссылка на модуль кода листа. Метод ProcStartLine возвращает номер пустой строки перед процедурой обработки события Combo_Click. Метод Delete удаляет четыре строки, включая одну пустую.

Несложно заметить, что динамическое создание элементов управления ActiveX требует определенных трудозатрат. Если дополнительная функциональность элементов управления ActiveX не требуется, проще использовать элементы управления с панели инструментов **Формы** (Forms).

Элементы управления, встроенные в диаграмму

На рис. 10.4 показана диаграмма, на которой расположена кнопка для удаления или добавления последовательности значений прибыли. Последовательность значений основана на значениях Планировщик прибыли на листе Profit. Элемент управления Button доступен на панели инструментов Формы (Forms). Этот элемент управления является частью коллекции Buttons (элементы управления ActiveX нельзя использовать на диаграммах).

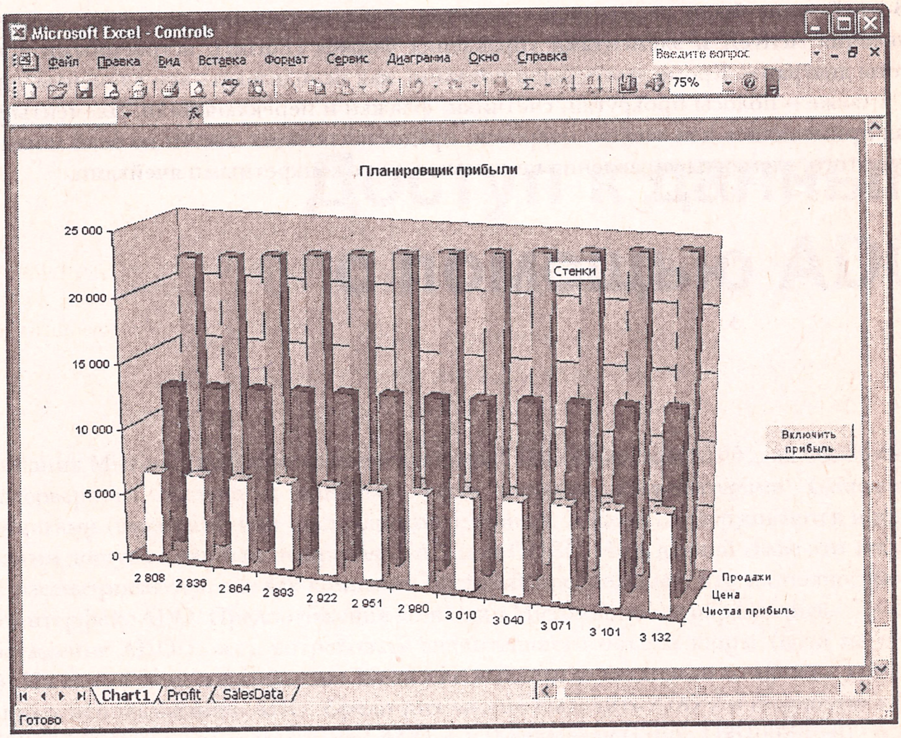


Рис. 10.4. Диаграмма с возможностью добавления и удаления рядов

Свойству OnAction объекта Button присваивается ссылка на следующий код:

```
Sub Button1_Click()
    With ActiveChart
        If .SeriesCollection.Count = 3 Then
            .SeriesCollection(1).Delete
        Else
            With .SeriesCollection.NewSeries
                .Name = Sheet1.Range("A13")
                .Values = Sheet1.Range("B13:M13")
                .XValues = Sheet1.Range("B12:M12")
                .PlotOrder = 1
            End With
        End If
    End With
End Sub
```

Если свойству `SeriesCollection.Count` присвоить значение 3, первый ряд удаляется. В противном случае добавляется новый ряд, который связывается с соответствующими диапазонами значений прибыли, отображаемыми после значений налогов. Новый ряд добавляется последним и должен отображаться за существующими рядами. Для отображения нового ряда перед существующими свойству `PlotOrder` присваивается соответствующее значение.

Резюме

В этой главе были рассмотрены отличия между встроенными в лист элементами управления ActiveX и элементами управления **Формы** (Forms), встроенными в листы и листы диаграмм. Кроме этого, было показано, как использовать эти элементы управления, а также — полосы прокрутки, счетчики, флажки и переключатели. Элементы управления применялись для запуска макросов, предоставляющих доступ ко всей мощи VBA. Кроме этого, элементы управления не связываются с конкретными ячейками.

11

Доступ к данным с помощью ADO

Компания Microsoft выбрала технологию *ActiveX Data Object*, или ADO, для обеспечения клиент-серверного доступа к данным между любыми потребителями (клиентами) и источниками (серверами или поставщиками) данных. В Excel поддерживаются и другие технологии доступа к данным, например DAO или ODBC. Но в данной главе эти технологии не рассматриваются, так как компания Microsoft предполагает, что их полностью вытеснит интерфейс ADO. Предположения компании Microsoft почти оправдались.

Обсуждение ADO может потребовать значительного объема книги (если не целую книгу). На самом деле издательство Wrox уже выпустило несколько отличных книг, посвященных использованию ADO. Среди них можно выделить *ADO 2.6 Programmer's Reference* (ISBN 1-861004-63-x) и *Professional ADO 2.5 Programming* (ISBN 1-861002-75-0). В этой главе рассматривается небольшое подмножество возможностей технологии ADO и описываются ситуации, которые часто возникают при программировании приложений для Excel. Дополнительная информация об ADO доступна в одной из упомянутых ранее книг.

Как отдельная универсальная технология доступа к данным, интерфейс ADO быстро развивался в течение последних нескольких лет. Развитие технологии происходило намного быстрее, чем развитие использующих технологию приложений. На момент написания настоящей книги широко применяются несколько версий технологии ADO. Это версии 2.1, 2.5, 2.6 и 2.7. В этой главе рассматривается версия ADO 2.7. Она предоставляется вместе с последними версиями Windows и Office. Если ни одно из этих приложений не используется и библиотека ADO 2.7 не установлена, ее можно загрузить с сайта Microsoft Universal Data Access, который доступен по адресу `http://www.microsoft.com/data`.

Введение в структурированный язык запросов

Обсуждение доступа к данным невозможно без рассмотрения SQL. SQL — это язык запросов, используемый для общения со всеми распространенными базами данных. Язык SQL основан на стандартах, в которых существует столько же вариантов, сколько существует производителей баз данных. В этой главе рассматриваются конструкции, по возможности совместимые со стандартом SQL-92. Но при рассмотрении доступа к данным с применением Microsoft SQL Server будет использоваться вариант языка SQL, который называется Transact SQL или T-SQL.

В этой главе приводится краткое описание базового синтаксиса SQL. Обзор ни в коем случае не является полным, но его будет достаточно для понимания основных концепций, которые используются в этой главе. Дополнительная информация с примерами применения языка SQL доступна в книге *Beginning SQL Programming* (ISBN 1-861001-80-0) издательства Wrox Press.

Ниже перечислены четыре команды SQL, используемые чаще всего. Кроме этого, указана пара команд, применяемых не так часто, но являющихся достаточно мощными, чтобы указать их здесь. Вот эти команды:

- `SELECT` — используется для получения данных из источника;
- `INSERT` — используется для добавления записей в источник;
- `UPDATE` — используется для модификации существующих записей в источнике;
- `DELETE` — используется для удаления записей из источника;
- `CREATE TABLE` — используется для создания новой таблицы;
- `DROP TABLE` — используется для удаления существующей таблицы.

Термины *запись (record)* и *поле (field)* часто применяются при описании данных. Источник данных будет рассматриваться в этой главе и может восприниматься в виде двумерной таблицы. Запись соответствует строке таблицы, а поле представляет столбец таблицы. Пересечение записи и поля представляют собой *значение (value)*. *Результирующее множество (resultset)* описывает возвращаемое множество данных, полученное в результате выполнения оператора `SELECT`.

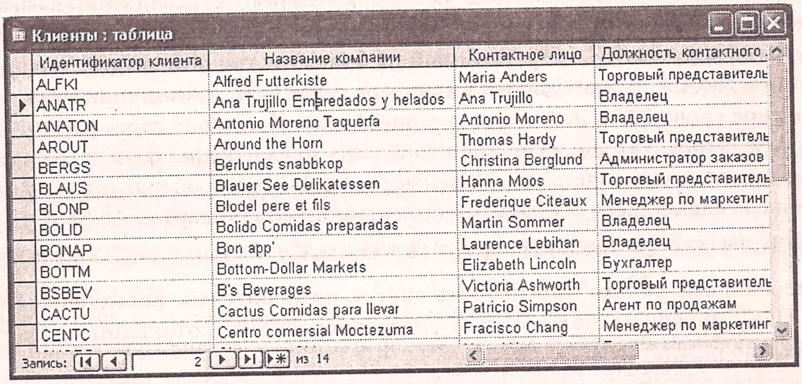
> Можно обратить внимание, что ключевые слова SQL, например `SELECT` и `UPDATE`, указываются в верхнем регистре. Это распространенная практика программирования на языке SQL. При просмотре сложных операторов SQL использование верхнего регистра для записи ключевых слов позволяет отличать их от операндов. Фрагменты операторов SQL называются предложениями. Во всех операторах SQL некоторые предложения являются обязательными, а некоторые — необязательными. При описании синтаксиса операторов SQL необязательные предложения и ключевые слова заключаются в квадратные скобки.

Для демонстрации примеров кода SQL будет использоваться таблица `Customers` из демонстрационной базы данных Microsoft `Northwood` (рис. 11.1). База данных `Northwood` устанавливается вместе с пакетом Microsoft Access или при установке Microsoft SQL Server (обычно этот файл называется `NWind.mdb` или `Northwood.mdb`).

Оператор SELECT

Оператор SELECT является самым распространенным оператором языка SQL, позволяющим извлекать данные из источника. Базовый синтаксис оператора SELECT выглядит следующим образом:

```
SELECT столбец1, столбец2 [, столбец_n] FROM таблица
```



Клиенты : таблица

Идентификатор клиента	Название компании	Контактное лицо	Должность контактного
ALFKI	Alfred Futterkiste	Maria Anders	Торговый представитель
ANATR	Ana Trujillo Emparedados y helados	Ana Trujillo	Владелец
ANATON	Antonio Moreno Taquerfa	Antonio Moreno	Владелец
AROUT	Around the Horn	Thomas Hardy	Торговый представитель
BERGS	Berlunds snabbkop	Christina Berglund	Администратор заказов
BLAUS	Blauer See Delikatessen	Hanna Moos	Торговый представитель
BLONP	Blodel pere et fils	Frederique Citeaux	Менеджер по маркетинг
BOLID	Bolido Comidas preparadas	Martin Sommer	Владелец
BONAP	Bon app'	Laurence Lebihan	Владелец
BOTTM	Bottom-Dollar Markets	Elizabeth Lincoln	Бухгалтер
BSBEV	B's Beverages	Victoria Ashworth	Торговый представитель
CACTU	Cactus Comidas para llevar	Patricio Simpson	Агент по продажам
CENTC	Centro comersial Moctezuma	Fracisco Chang	Менеджер по маркетинг

Запись: 2 из 14

Рис. 11.1. Таблица Customers

В показанной на рис. 11.1 таблице содержится столбец, который называется CompanyName. Для выбора имен всех потребителей можно написать следующий оператор:

```
SELECT CompanyName FROM Customers
```

Предложение SELECT сообщает источнику данных о столбцах, которые необходимо извлечь. Предложение FROM сообщает источнику данных имя таблицы, хранящей интересующие записи. Пример можно расширить и выбрать не только имя компании, но и имя контактного лица. Для этого можно воспользоваться следующим запросом:

```
SELECT CompanyName, ContactName FROM Customers
```

Оператор сообщает источнику данных о том, что необходимо получить все значения полей CompanyName и ContactName из таблицы Customers. Оператор SELECT предоставляет короткий вариант записи запроса для получения всех полей из указанной таблицы. Для этого в операторе SELECT необходимо указать один символ *:

```
SELECT *FROM Customers
```

В результате выполнения этого оператора SQL будут предоставлены все поля и все записи из таблицы Customers. Обычно не рекомендуется использовать символ * в операторах SELECT, так как код становится уязвим по отношению к изменениям имен или порядка полей. Кроме этого, обработка такого запроса может потребовать слишком много ресурсов при наличии таблиц большого объема, так как возвращаются все записи, даже те, которые клиенту не нужны. Но иногда возникают ситуации, когда такая возможность может оказаться полезной.

Предположим, что необходимо просмотреть список стран, в которых есть как минимум один потребитель. Выполнение следующего запроса позволит получить по одной записи для каждого потребителя в таблице.

```
SELECT Country FROM Customers
```

В результате этого запроса будет присутствовать много дублированных имен стран. Необязательное ключевое слово DISTINCT позволяет получить в результате запроса только уникальные значения:

```
SELECT DISTINCT Country FROM Customers
```

Ключевое слово DISTINCT может не поддерживаться некоторыми производителями баз данных. Точный синтаксис и ключевые слова конкретной реализации языка SQL доступны в документации, которую предоставляет каждый производитель баз данных. Если необходимо создать код SQL, переносимый на большинство реализаций, воспользуйтесь предложением GROUP BY, входящим в стандарт ANSI SQL и выполняющим ту же роль, что и ключевое слово DISTINCT в реализации Access. Ключевое слово GROUP BY позволяет добиться, чтобы каждое значение столбца из предложения GROUP BY присутствовало в результате только один раз. Поведение предыдущего оператора можно реализовать с помощью следующего оператора, совместимого со стандартом ANSI SQL:

```
SELECT Country FROM Customers GROUP BY Country
```

Если необходимо только просмотреть список потребителей, находящихся в Великобритании (UK), можно воспользоваться оператором WHERE, который позволяет ограничить множество результатов:

```
SELECT CompanyName, ContactName
FROM Customers
WHERE Country = 'UK'
```

Обратите внимание, что строка **UK** должна быть заключена в одинарные кавычки. Это касается и дат. Числовые выражения в кавычки заключать не нужно.

Наконец, предположим, что необходимо отсортировать список потребителей из Великобритании по значению поля CompanyName. Для этого можно воспользоваться предложением ORDER BY:

```
SELECT CompanyName, ContactName
FROM Customers
WHERE Country = 'UK'
ORDER BY CompanyName
```

По умолчанию при использовании предложения ORDER BY выполняется сортировка по возрастанию. Если вместо этого поля необходимо отсортировать по убыванию, можно воспользоваться необязательным квалификатором DESC. Его необходимо указать сразу после имени столбца, порядок сортировки которого нужно изменить.

Оператор INSERT

Оператор INSERT позволяет добавлять в таблицу новые записи. Базовый синтаксис оператора INSERT выглядит следующим образом:

```
INSERT INTO имя_таблицы (столбец1, столбец2 [, столбец_n])
VALUES (значение1 , значение2 [, значение_n])
```

Оператор INSERT очень прост в применении. Нужно указать имя таблицы и столбцов, в которые необходимо вставить данные. Кроме этого, предоставить вставляемые значения. Предоставляемые значения указываются в предложении VALUES. Необходимо указать значение для каждого столбца, имя которого находится в предложении INSERT.

Значения должны предоставляться в том же порядке, в котором перечислены имена столбцов. Вот пример вставки новой записи в таблицу Customers:

```
INSERT INTO Customers (CustomerID, CompanyName, ContactName, Country)
VALUES ('ABCD', 'New Company', 'Owner Name', 'USA')
```

Обратите внимание, что как и в случае с предложением WHERE в операторе SELECT, все строковые значения в предложении VALUES заключаются в одинарные кавычки. Это правило действует для всех операторов языка SQL.

Если значения предоставляются для каждого поля в таблице и перечислены в том же порядке, что и поля, предложение с перечислением полей можно опустить:

```
INSERT INTO Customers
VALUES( 'ALFKJ', 'Alfreds Futterkiste', 'Maria Anders', 'Sales _
  Representative',
'Obere Str. 57', 'Berlin', 'Hessen', '12209', 'Germany', ' 030-0074321', _
'030-0076545')
```

Оператор UPDATE

Оператор UPDATE позволяет модифицировать значения одного или более полей существующей записи или записей таблицы. Базовый синтаксис оператора UPDATE выглядит следующим образом:

```
UPDATE имя_таблицы
SET столбец1 = значение1, столбец2 = значение2
↳ [,столбец_n = значение_n]
[WHERE фильтры]
```

Хотя предложение WHERE оператора UPDATE является необязательным, его необходимо указывать, кроме тех случаев, когда точно известно, что оно не потребуется. Выполнение оператора UPDATE без предложения WHERE приведет к модификации указанных полей всех записей в указанной таблице. Например, если выполнить следующий оператор:

```
UPDATE Customers
SET Country = 'USA'
```

то в каждой записи таблицы Customers значение поля Country будет установлено равным "USA". Существуют ситуации, когда возможность массовой модификации записей может оказаться полезной, но она же может оказаться и опасной, так как нельзя отменить обновление, запущенное по ошибке. Следовательно, при разработке запросов лучше создать резервную копию базы данных и экспериментировать на специально созданном наборе данных, что позволит восстановить базу данных из резервной копии в случае повреждения в результате выполнения неправильного запроса.

Чаще всего оператор UPDATE используется для модификации значения в конкретной записи, на которую указывает предложение WHERE. Перед рассмотрением примера стоит обсудить один очень важный аспект проектирования баз данных: *первичный ключ (primary key)*. Первичный ключ представляет собой столбец или группу столбцов, уникально идентифицирующих каждую запись в таблице. В базе данных Customers в качестве первичного ключа используется столбец CustomerID. В каждой записи таблицы Customers присутствует уникальное значение поля CustomerID. Другими словами, конкретное значение поля CustomerID присутствует только в одной записи клиента в таблице.

Предположим, что для клиента “Around the Horn” изменилось имя контактного лица. Значение поля CustomerID для этого клиента равно "AROUT". Для изменения записи для этого клиента можно воспользоваться следующим оператором UPDATE:

```
UPDATE Customers
SET ContactName = 'Новое название'
WHERE CustomerID = 'AROUT'
```

Так как в оператор добавлен предикат WHERE, который основан на первичном ключе и обеспечивает уникальность записи, то будет обновлена только одна запись — запись клиента “Around the Horn”.

Оператор DELETE позволяет удалять из таблицы одну или несколько записей. Базовый синтаксис оператора DELETE выглядит следующим образом:

```
DELETE FROM имя_таблицы
[WHERE фильтр]
```

Как и в случае с оператором UPDATE, предложение WHERE является необязательным. Отказ от использования этого предложения в строке оператора DELETE может показаться еще более опасным, так как применение оператора DELETE без предложения WHERE приведет к удалению каждой записи в указанной таблице. Если не планируется удалять все строки, лучше всегда добавлять предложение WHERE в операторы DELETE. И в данном случае стоит отметить, что разработку запросов имеет смысл выполнять на специально созданном наборе данных. Например, при работе с базой данных Access достаточно сделать копию файла .mdb и работать с ней. Предположим, что по какой-то причине в таблицу Customers внесена запись с полем CustomerID, равным "BONAP" (возможно, это поставщик, а не клиент). Для удаления этой записи из таблицы Customers можно воспользоваться следующим оператором DELETE:

```
DELETE FROM Customers
WHERE CustomerID = 'BONAP'
```

Так как в предложении WHERE используется значение основного ключа записи, удаляется только одна запись.

Оператор CREATE TABLE

Оператор CREATE TABLE применяется для создания новых таблиц в существующей базе данных. Хотя этот оператор, скорее всего, будет использоваться намного реже, чем операторы SELECT, UPDATE, INSERT и DELETE, с его возможностями стоит познакомиться. Оператор CREATE TABLE можно применять для предоставления потребителям возможности создавать собственные таблицы после передачи готового решения. Кроме этого, оператор CREATE TABLE может использоваться для создания резервных таблиц или для репликации существующей таблицы в новую. Оператор CREATE TABLE имеет следующий синтаксис:

```
CREATE TABLE имя_таблицы (имя_столбца1 тип1 [,имя_столбца_n тип_n])
```

Помните, что обычно в таблицах присутствует несколько важных элементов. Одним из них является первичный ключ, вторым — сущность, которую иногда называют внешним ключом. Внешний ключ содержит уникальное значение, используемое в качестве индекса в другой таблице. Например, предположим, что необходимо расширить таблицу Employees в базе данных Northwood, не меняя содержимое таблицы. Для этого можно

создать новую таблицу с собственным первичным ключом и уникальным столбцом EmployeeID, который логически связан со столбцом Emploeeys.EmployeeID. Наконец, можно добавить дополнительную информацию, например, адрес электронной почты. Такой запрос будет выглядеть следующим образом:

```
CREATE TABLE EmployeeContactInformation

(EmployeeContactInformationID Int Primary Key,
  EmployeeID Int UNIQUE,
 Email Text)
```

После выполнения этого запроса в базе данных Northwood будет получена таблица EmployeeContactInformation с первичным ключом EmployeeContactInformationID, уникальным (без повторений) столбцом EmployeeID, представляющим логическое отношение между новой таблицей и таблицей Employees, и дополнительным полем Email, используемым для хранения адреса электронной почты сотрудника.

Очевидно, что можно модифицировать таблицу Employees и добавить соответствующее поле, но такое решение не всегда является желательным. Например, если база данных предоставляется внешним источником, такое изменение может привести к нарушению работы существующих приложений.

Оператор DROP TABLE

Первыми программистами были математики. Математики предпочитают использовать симметричную терминологию. Таким образом, если существует оператор CREATE TABLE, очевидно, должен существовать и оператор для удаления таблиц. Оператор удаления таблицы (целой таблицы, а не всех строк в таблице) выглядит следующим образом:

```
DROP TABLE имя_таблицы
```

Предположим, что таблица EmployeeContactInformation создавалась в качестве временного решения. Тогда следующий оператор позволяет удалить эту таблицу:

```
DROP TABLE EmployeeContactInformation
```

Обзор технологии ADO

ADO является универсальной технологией доступа к данным от компании Microsoft. Под универсальностью подразумевается проектирование ADO для предоставления доступа к любому вообразимому источнику данных. Это может быть база данных SQL Server, база данных Windows 2000 Active Directory, текстовый файл на локальном жестком диске и даже продукт не от компании Microsoft, например база данных Oracle. Ко всем этим источникам данных можно получить доступ с помощью интерфейсов ADO. Большой объем информации об ADO доступен в разделе ADO на сайте Microsoft Universal Data Access по адресу http://www.microsoft.com/data/ado/.

Интерфейс ADO не пытается обратиться непосредственно к источнику данных. Вместо этого ADO является потребителем данных, получаемых через технологию более низкого уровня, которая называется *OLE DB*. Доступ к OLE DB непосредственно из VBA невозможен, поэтому для этих целей используется технология ADO. ADO получает данные от поставщиков OLE DB (OLE DB providers). Большинство поставщиков OLE DB предназначены для работы с единственным источником данных. Каждый поставщик предназначен для предоставления общего интерфейса к любому содержимому источника дан-

ных. Одним из основных преимуществ ADO является использование одного независимого от источника данных набора команд. Это позволяет не изучать новые технологии или методы при доступе к разным источникам данных.

Кроме этого, компания Microsoft предоставляет поставщика OLE DB для интерфейса ODBC. Этот универсальный поставщик позволяет ADO получать доступ к любому источнику данных, который поддерживает использование интерфейса ODBC, даже если для этого источника не существует специализированного поставщика данных OLE DB.

Объектная модель ADO состоит из пяти объектов верхнего уровня, каждый из которых можно создавать независимо от остальных. В этой главе рассматриваются объекты `Connection`, `Command` и `Recordset`. Кроме этого, ADO предоставляет объект `Record` (не путайте его с объектом `Recordset`) и объект `Stream`. Эти объекты достаточно редко используются в приложениях Excel, поэтому интересующиеся читатели могут обратиться к источникам, перечисленным в начале этой главы.

Кроме пяти объектов верхнего уровня, ADO предоставляет четыре коллекции и объекты, которые входят в эти коллекции (например, коллекция `Errors` содержит объекты `Error`). Научитесь использовать несколько классов и можете считать, что знаете ADO. Объектная модель ADO показана на рис. 11.2.

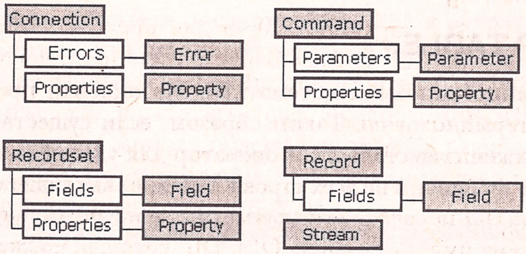


Рис. 11.2. Фрагмент объектной модели ADO

В следующих трех разделах рассматривается каждый используемый в этой главе объект верхнего уровня из объектной модели ADO. В этих разделах предоставляется общая информация, необходимая для эффективного применения ADO. Конкретные примеры использования ADO для решения задач доступа к данным средствами Excel VBA рассматриваются в следующих разделах.

Не стоит воспринимать эту главу, как подробный справочник по ADO. Здесь рассматриваются только те компоненты, которые будут использоваться в этой главе, или компоненты, на которые стоит обратить внимание. Часто ADO предоставляет возможность установки параметра несколькими способами, например в качестве свойства объекта или в качестве аргумента метода. В таком случае будет рассматриваться метод, демонстрируемый в разделе с примерами.

Объект Connection

Объект `Connection` используется для создания конвейера между приложением и источником данных, к которому получает доступ приложение. Как и другие объекты ADO верхнего уровня, объект `Connection` предоставляет исключительную гибкость. В некоторых случаях достаточно использовать только этот объект. Простые команды могут выполняться непосредственно через объект `Connection`. Иногда вообще можно обойтись

без создания объекта Connection. Объекты Command и Recordset могут создавать необходимый объект Connection автоматически.

Создание и удаление подключения к источнику данных может потребовать определенных затрат времени. Если в процессе работы приложения планируется выполнять несколько операторов SQL, стоит создать глобальную объектную переменную Connection и использовать ее для выполнения каждого запроса. Это позволит воспользоваться возможностями *пула подключений (connection pooling)*.

Пул подключений предоставляется технологией ADO. Эта возможность позволяет сохранять и повторно использовать подключение к источнику данных, не требуя создания нового подключения для каждого запроса (многократное создание запросов потребовало бы значительных ресурсов). Подключение может повторно применяться даже для разных запросов. Достаточно, чтобы были одинаковы строки подключения. Такая ситуация характерна для приложений Excel, поэтому можно порекомендовать воспользоваться этой возможностью.

Свойства объекта Connection

В этом разделе рассматриваются важные свойства объекта Connection.

Свойство ConnectionString

Свойство ConnectionString применяется для предоставления интерфейсу ADO и используемому поставщику OLE DB необходимой для подключения к источнику данных информации. Строка подключения состоит из последовательности аргументов, разделенных точкой с запятой. Аргументы имеют форму "имя=значение;".

В этой главе из аргументов ADO будет использоваться только аргумент Provider. Все остальные аргументы строки подключения относятся к конкретному поставщику OLE DB. Интерфейс ADO передает эти аргументы непосредственно поставщику. Аргумент Provider содержит имя поставщика OLE DB, который должен использоваться интерфейсом ADO. В следующем примере кода показана строка подключения к базе данных Northwind с помощью поставщика Jet OLE DB:

```
"Provider=Microsoft.Jet.OLEDB.4.0;Data Source=C:\Program Files\Microsoft
Office\OFFICE11\SAMPLES\Northwind.mdb;Persist Security Info=False"
```

Единственным аргументом интерфейса ADO является аргумент Provider. Все остальные аргументы передаются непосредственно поставщику SQL Server OLE DB. Если используется другой поставщик, аргументы также будут отличаться. В этом можно убедиться при подключении к другим источникам данных в разделе примеров. Аргумент Provider в строке подключения можно не указывать. Если не указать поставщика, интерфейс ADO по умолчанию будет использовать поставщика OLE DB ODBC.

Кого-то может удивить сложность строки подключения. Как можно запомнить сложную строку текста, особенно если аргументы зависят от используемого поставщика OLE DB? Именно для этого покупаются такие книги, как эта.

Вместо того чтобы запоминать особенности каждого поставщика OLE DB, воспользуйтесь методикой для создания собственной строки подключения. Поставщик OLE DB реализован в библиотеке oledb32.dll. Эта динамически подключаемая библиотека предоставляет утилиту с графическим интерфейсом, в которой есть диалоговое окно Data Link Properties (Свойства подключения к данным) (рис. 11.3). Если создать пустой текстовый файл и изменить его расширение с .txt на .udl, то двойной щелчок на файле приведет к открытию диалогового окна Data Link Properties (Свойства подключения

к данным), так как расширение .udl связано с основными службами поставщика OLE DB. Одна из этих служб используется для определения подключения к поставщикам данных.

Диалоговое окно Data Link Properties (Свойства подключения к данным) является мастером. Активизируйте первую вкладку Поставщик (Provider), выберите поставщика и щелкните на кнопке Далее (Next). Следующая вкладка называется Подключение (Connection). Содержимое этой вкладки меняется в зависимости от выбора на первой вкладке (каждый поставщик предоставляет различные параметры). В данном случае выберите поставщика Microsoft Jet 4.0 OLE DB Provider, MS Access OLE DB. На вкладке Подключение (Connection) достаточно найти и выбрать файл Northwind.mdb, и работу мастера можно считать законченной. После щелчка на кнопке OK диалоговое окно закрывается и в файл .udl записывается строка подключения. Откройте файл .udl с помощью редактора Блокнот (Notepad) и скопируйте строку подключения. На рис. 11.4 показано содержимое файла .udl, открытого в редакторе Блокнот (Notepad). Интерес представляет последняя строка текста; первые две содержат комментарий.

Вот код, который можно использовать для инициализации объекта подключения ADO и присвоения строки подключения свойству ConnectionString объекта подключения. (Не забудьте воспользоваться командой меню Сервис⇨Ссылки (Tools⇨References) в редакторе VBE для добавления ссылки на библиотеку Microsoft ActiveX Data Objects 2.7 Library (ADO).)

```
Const ConnectionString As String = _
  "Provider=Microsoft.Jet.OLEDB.4.0;" + _
  "Data Source=C:\Program Files\Microsoft " + _
  "Office\OFFICE11\SAMPLES\Northwind.mdb;Persist Security Info=False"
Dim Connection As ADODB.Connection
Set Connection = New ADODB.Connection
Connection.ConnectionString = ConnectionString
```

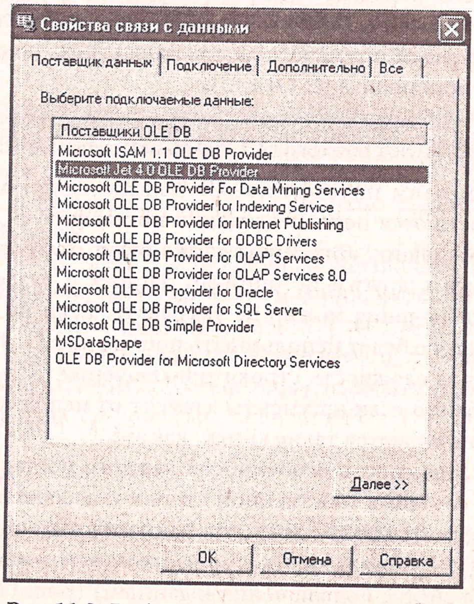


Рис. 11.3. Выбор поставщика OLE DB

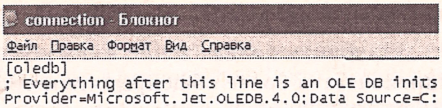


Рис. 11.4. Строка подключения к базе данных

Первый оператор определяет константную переменную, инициализированную строкой подключения, которая скопирована из файла `.udl`. Второй оператор объявляет объект подключения. После этого объект подключения инициализируется и свойству `ConnectionString` присваивается строка подключения.

Свойство ConnectionTimeout

Практически во всех классах существует несколько членов. Класс `Connection` предоставляет свойство `ConnectionTimeout` и определяет период, в течение которого ADO будет ожидать установки подключения перед выдачей сообщения об ошибке. По умолчанию этот период составляет 15 секунд. Изменение значения этого свойства требуется редко, так как подключение или устанавливается, или нет. Ниже приводится код для модификации значения свойства `ConnectionTimeout`.

```
Connection.ConnectionTimeout = 30
```

Свойство State

Свойство `State` позволяет определить текущее состояние подключения. Подключение может быть установлено или разорвано в процессе подключения или выполнения команды. Значение свойства представляет собой битовую маску, в которой содержится одна или несколько констант `ObjectStateEnum`, определенных в библиотеке ADO:

- `AdStateClosed` — подключение разорвано;
- `AdStateOpen` — подключение установлено;
- `AdStateConnecting` — подключение устанавливается;
- `AdStateExecuting` — подключение выполняет команду.

При попытке разорвать уже разорванное подключение или открыть уже открытое будет выдано сообщение об ошибке. Для предотвращения такой ситуации можно проверить состояние объекта `Connection` перед использованием:

```
If Connection.State = ObjectStateEnum.adStateOpen) Then objConn.Close
```

На первый взгляд код, пытающийся установить уже установленное подключение, может показаться смешным, но попытки установить подключение могут оказаться неудачными или подключение может быть полем другого класса и другой код может попытаться установить или разорвать существующее подключение. Для предотвращения такой ситуации всегда проверяйте состояние подключения перед вызовом методов `Open` или `Close`.

Методы объекта Connection

Методы определяют поведение объектов. Объект `Connection` предоставляет подключение к поставщику. Не сложно представить, что подключения могут быть установлены или разорваны, но библиотека ADO предоставляет дополнительные методы, которые могут оказаться полезными.

Метод Open

Метод Open позволяет установить подключение к поставщику (так называемому источнику данных; поставщик является более общепринятым термином). Этот метод принимает несколько необязательных параметров. Если предварительно инициализировать свойство ConnectionString, то метод Open можно вызывать без параметров. Если свойство ConnectionString не инициализировано, строку подключения, идентификатор пользователя, пароль и дополнительный аргумент Options можно передавать в качестве параметра метода. Все эти параметры рассматриваются ниже. В следующем примере показано, как инициализировать объект, устанавливать подключение и проверять состояние подключения:

```
Const ConnectionString As String = _
  "Provider=Microsoft.Jet.OLEDB.4.0;" + _
  "Data Source=C:\bs Program Files\bs Microsoft " + _
  "Office\bs OFFICE11\bs SAMPLES\bs Northwind.mdb;Persist
↳ Security Info=False"

Dim Connection As ADODB.Connection
Set Connection = New ADODB.Connection
Connection.ConnectionString = ConnectionString
Connection.Open
MsgBox Connection.State = ObjectStateEnum.adStateOpen
```

Выше упоминался дополнительный аргумент метода Options. Аргумент Options позволяет сделать установленное подключение *асинхронным (asynchronous)*. То есть, объект Connection можно заставить сразу вернуть управление и устанавливать подключение в фоновом режиме одновременно с работой другого кода. Для этого аргумент Options необходимо установить в значение adAsyncConnect из перечислимого типа ConnectOptionEnum. Следующий фрагмент кода позволяет установить асинхронное подключение:

```
objConn.Open Options:=adAsyncConnect
```

Такая возможность может оказаться полезной для инициализации подключения в начале работы кода. Пока подключение будет устанавливаться в фоновом режиме, остальной код может выполнять другие задачи.

Метод Execute

Метод Execute выполняет команду, текст которой предоставляется в качестве значения аргумента CommandText. Метод Execute использует следующий синтаксис для запроса на выполнение операции (запрос не должен возвращать результирующее множество, например, DELETE, INSERT или UPDATE; SELECT к таким запросам не относится):

```
connection.Execute CommandText, [RecordsAffected], [Options]
```

Для запроса на получение данных возвращаемое значение метода Execute присваивается объекту Recordset:

```
Set Recordset = connection.Execute(CommandText, _
                                   [RecordsAffected], [Options])
```

Аргумент CommandText может содержать любую строку команды, воспринимаемую поставщиком OLE DB в качестве оператора SQL. Необязательный аргумент RecordsAffected содержит возвращаемое значение, позволяющее определить количество записей, затронутых выполнением команды. Это значение можно сравнить с ожидаемым количеством записей, что позволит обнаружить потенциальные ошибки в тексте команды.

Аргумент Options является важным элементом оптимизации эффективности работы команды. Таким образом, стоит всегда использовать этот аргумент, хотя формально он и является необязательным. Аргумент Options позволяет передавать поставщику OLE DB два информационных сообщения: поставщик получает информацию о типе команды, которая передается в качестве значения аргумента CommandText, а также получает рекомендацию по выполнению содержимого аргумента CommandText.

Для выполнения команды из аргумента CommandText поставщик OLE DB должен знать тип выполняемой команды. Если не указать тип, поставщик определит его самостоятельно. При этом выполнение запроса замедлится. Этого замедления можно избежать, если указать тип команды в CommandText с помощью одного из следующих параметров CommandTypeEnum:

- AdCmdText — значение аргумента CommandText является простой строкой SQL;
- AdCmdTable — аргумент CommandText содержит имя таблицы. При этом поставщику отправляется сгенерированный оператор SQL, который можно расшифровать как "SELECT * FROM *имя_таблицы*";
- AdCmdStoredProc — аргумент CommandText содержит имя хранимой процедуры (хранимые процедуры будут рассматриваться в разделе, посвященном SQL Server);
- AdCmdTableDirect — аргумент CommandText содержит имя таблицы. В отличие от AdCmdTable, в этом случае не генерируется оператор SQL и содержимое таблицы возвращается более эффективно. Если поставщик поддерживает этот параметр, стоит использовать именно его.

Поставщику можно передать конкретные инструкции по исполнению, указав одну или несколько констант ExecuteOptionEnum:

- AdAsyncExecute — от поставщика требуется асинхронное исполнение команды. Код приложения получает управление немедленно после отправки команды;
- AdExecuteNoRecords — поставщику запрещается создание объекта Recordset. Библиотека ADO всегда создает объект Recordset в результате выполнения команды (даже если команда CommandText не подразумевает возврата массива строк). Этот параметр стоит использовать для экономии на накладных расходах при создании ненужного объекта Recordset, когда запрос не предполагает возврата строк с данными.

Значения CommandTypeEnum и ExecuteOptionEnum являются битовыми масками, которые комбинируются в аргументе Options с помощью логического оператора ИЛИ. Например, для выполнения простой текстовой команды SQL и возврата объекта Recordset можно воспользоваться следующим кодом:

```
Const ConnectionString As String = _
  "Provider=Microsoft.Jet.OLEDB.4.0;" + _
  "Data Source=C:\Program Files\Microsoft " + _
  "Office\OFFICE11\SAMPLES\Northwind.mdb;Persist Security Info=False"

Dim Connection As ADODB.Connection
Set Connection = New ADODB.Connection
Connection.ConnectionString = ConnectionString
Connection.Open

Debug.Print Connection.State = ObjectStateEnum.adStateOpen
```

```
Const SQL As String = _
  "SELECT * FROM Customers WHERE Country = 'USA'"

Dim Recordset As Recordset
Dim RowsAffected As Long
Set Recordset = Connection.Execute(SQL, RowsAffected, Com-
mandTypeEnum.adCmdText)
```

Этот код основан на нашем примере. Вместо оператора MsgBox используется вызов метода Debug.Print, так как появление диалогового окна не должно прерывать работу кода, но интерес представляет информация о текущем состоянии приложения. После вызова метода Debug.Print определяется оператор SQL, переменная Recordset и переменная для хранения длинного целого числа, в которой будет храниться количество полученных строк. (Если количество возвращаемых строк не интересует, аргумент RowsAffected можно опустить и вместо него ввести дополнительную запятую, как показано ниже.)

```
Set Recordset = Connection.Execute(SQL, , adCmdText)
```

Метод Close

После завершения работы с поставщиком необходимо разорвать подключение. Существует конечное количество дескрипторов для подключения к базе данных, поэтому если не разрывать ненужные подключения, рано или поздно не удастся установить новое подключение к базе данных из других приложений. Метод Close не требует передачи аргументов. Комбинация из проверки текущего состояния подключения и вызова метода Close может выглядеть следующим образом:

```
If (Connection.State = ObjectStateEnum.adStateOpen) Then
  Connection.Close
End If
```

События объекта Connection и асинхронное программирование

Синхронность означает, что операции выполняются одна за другой. Асинхронность означает, что операции выполняются в любом порядке. Мощность асинхронного программирования заключается в возможности осуществлять фоновые операции одновременно с операциями, которые выполняются на переднем плане. Например, если запрос требует некоторого времени для выполнения, во время его реализации пользователь может выполнять другие задачи.

Обычно асинхронный вызов сразу возвращает управление. Возврат управления из асинхронного вызова не означает, что обработка вызова завершена. На самом деле обработка выполняется в фоновом потоке (современные центральные процессоры и операционные системы Windows поддерживают одновременную работу нескольких потоков; это позволяет в одно и то же время слушать музыку в Windows Media Player, набирать текст в Word и отправлять запрос к базе данных Access), а код, отправивший запрос, продолжает работать на переднем плане. Существует множество способов уведомления о завершении работы фонового потока. В Excel поддерживается один из таких способов — события. Если связать событие с подключением, фоновый процесс может создать событие после установки подключения, и основной код сможет продолжить работу. При правильном использовании асинхронное поведение и события позволяют увеличить произ-

водительность быстродействия. В результате уменьшаются видимые задержки и увеличивается скорость реакции приложения.

Для связи события с объектом `Connection` необходимо воспользоваться оператором `WithEvents` в модуле класса при объявлении объекта подключения. После применения оператора `WithEvents` для выбора объекта подключения и доступных событий можно использовать раскрывающиеся списки Object (Объект) и Procedure (Процедура). Разработчику предоставляется несколько событий, включая `BeginTransComplete`, `CommitTransComplete`, `ConnectComplete`, `Disconnect`, `ExecuteComplete`, `InfoMessage`, `RollbackTransComplete`, `WillConnect` и `WillExecute`. Для реакции на завершение работы асинхронной команды необходимо реализовать обработку события `ExecuteComplete`. Следующий пример является переработкой уже показанного кода. В этой версии продемонстрированы асинхронная обработка запроса SQL и наполнение диапазона данными из объекта `Recordset`:

```
Option Explicit
Private WithEvents AsyncConnection As ADODB.Connection

Public Sub AsyncConnectionToDatabase()

  Const ConnectionString As String = _
     "Provider=Microsoft.Jet.OLEDB.4.0;" + _
     "Data Source=C:\Program Files\Microsoft " + _
     "Office\OFFICE11\SAMPLES\Northwind.mdb;" + _
     "Persist Security Info=False"

  Set AsyncConnection = New ADODB.Connection

  AsyncConnection.ConnectionString = ConnectionString
  AsyncConnection.Open

  Const SQL As String = _
     "SELECT * FROM Customers WHERE Country = 'USA'"

  Call AsyncConnection.Execute(SQL, , CommandTypeEnum.adCmdText _
     Or ExecuteOptionEnum.adAsyncExecute)

  Debug.Print "Запускается до завершения обработки запроса"

End Sub

Private Sub AsyncConnection_ExecuteComplete( _
  ByVal RecordsAffected As Long, _
  ByVal pError As ADODB.Error, _
  adStatus As ADODB.EventStatusEnum, _
  ByVal pCommand As ADODB.Command, _
  ByVal pRecordset As ADODB.Recordset, _
  ByVal pConnection As ADODB.Connection)

  Debug.Print "Query finished"

  If (adStatus = EventStatusEnum.adStatusOK) Then
     Call Sheet1.Range("A1").CopyFromRecordset(pRecordset)
  End If

  If (pConnection.State = ObjectStateEnum.adStateOpen) Then
     pConnection.Close
  End If
End Sub
```

Номера строк были добавлены для упрощения ориентации в листинге. При вводе кода в редакторе VBE номера строк не нужны. Асинхронные методы используются в двух операторах, а остальной код практически не отличается от предыдущего варианта.

Во второй строке объявляется оператор WithEvents, который позволяет генерировать процедуры обработки событий, в частности, метод AsyncConnection_ExecuteComplete. Асинхронный вызов метода Execute означает, что объект Recordset не передается в качестве возвращаемого значения и обработка результата работы метода выполняется в процедуре обработки события ExecuteComplete. Добавление ExecuteOptionEnum.adAsyncExecute в строке 21 делает вызов метода Execute асинхронным. Вызов метода Debug.Print в строке 23 добавлен для демонстрации выполнения кода до завершения обработки запроса.

Событие ExecuteComplete создается после завершения работы асинхронного метода Execute. Аргументы события ExecuteComplete позволяют определить, для какого подключения и команды создано событие. В данном примере проверяется завершение обработки запроса (строка 37). Если обработка запроса завершена, содержимое объекта Recordset копируется на лист в ячейку A1. В строке 41 проверяется состояние подключения. Если подключение установлено, оно разрывается.

Вызов метода Debug.Print в строке 23 можно заменить на другой код, который должен работать после вызова метода Execute. Например, при удалении таблицы операция удаления может выполняться в фоновом режиме, а пользователь при этом продолжать работу с приложением, практически не замечая снижения производительности.

Применение асинхронной обработки требует определенного внимания. Например, попытка разорвать подключение, используемое для фоновой обработки запроса, приведет к появлению сообщения об ошибке. К асинхронному программированию необходимо немного привыкнуть, но в результате такой прием позволяет добиться повышения производительности.

Коллекции объекта Connection

Объект Connection предоставляет две коллекции: Errors и Properties.

Коллекция Errors

В коллекции Errors содержится набор объектов Error. Каждый объект представляет ошибки, специфичные для разных поставщиков OLE DB (ошибки времени выполнения генерируются библиотекой ADO). В коллекции Errors могут храниться ошибки, предупреждения и сообщения (например, такие сообщения могут генерироваться оператором PRINT языка T-SQL), а также в ней может содержаться полезная информация для решения проблем в работе кода ADO. Следующий код позволяет вывести содержимое коллекции Errors в окно Immediate (Проверка).

```
Dim E As Error
  For Each E In pConnection.Errors
    Debug.Print E.Value
  Next
```

Коллекция Properties

В коллекции Properties хранятся специфичные для поставщиков расширенные свойства объекта Connection. Некоторые поставщики предоставляют важные параметры, о которых стоит знать разработчику. Расширенные свойства не рассматриваются в этой главе. Дополнительная информация о расширенных свойствах доступна в указанных выше справочных руководствах по библиотеке ADO.

Объект Recordset

Как оператор SELECT является самым распространенным оператором языка SQL, так и объект Recordset является самым часто используемым объектом библиотеки ADO. Объект Recordset выступает в роли контейнера для записей и полей, которые возвращаются в результате выполнения оператора SELECT источником данных.

Свойства объекта Recordset

Изучение любого класса подразумевает изучение свойств, описывающих состояние класса, методов, которые описывают поведение класса, и событий, определяющих ситуации, на которые может реагировать класс. Начнем изучение класса Recordset со свойства ActiveConnection.

Свойство ActiveConnection

Перед использованием объекта Recordset свойству ActiveConnection можно присвоить существующий объект Connection или строку подключения, которая будет применяться для подключения к базе данных. Если присвоить строку подключения, объект Recordset создаст объект Connection автоматически. После открытия объекта Recordset свойство ActiveConnection возвращает ссылку на объект Connection, который используется объектом Recordset.

Следующий фрагмент кода присваивает объект Connection свойству ActiveConnection:

```
Set Recordset.ActiveConnection = Connection
```

В этом фрагменте свойству ActiveConnection присваивается строка подключения, что приводит к неявному созданию объекта подключения:

```
Recordset.ActiveConnection = "Provider=Microsoft.Jet.OLEDB.4.0;" + _
        "Data Source=C:\Program Files\Microsoft " + _
        "Office\OFFICE11\SAMPLES\Northwind.mdb;" + _
                                "Persist Security Info=False"
```

Свойства BOF и EOF

Эти свойства указывают, находится ли указатель объекта Recordset перед первой записью в *наборе записей (recordset)* (BOF или начало файла) или после последней записи в наборе записей (EOF или конец файла). Если набор записей является пустым множеством, и BOF и EOF будут равны True. В следующем фрагменте кода показано использование этих свойств для определение наличия данных в наборе записей:

```
If (Recordset.EOF And Recordset.BOF) Then
    Debug.Print "Нет данных"
End If
```

Обратите внимание на разницу между пустым и закрытым наборами записей. Если в результате запроса не возвращаются данные, библиотека ADO предоставит полностью действительный объект Recordset, но не содержащий данных. Таким образом, даже если в результате выполнения запроса возвращается объект Recordset, стоит использовать вариант показанного кода для проверки наличия данных в этом объекте.

Кроме этого, объект Recordset предоставляет свойство RecordCount, но это свойство поддерживается не всеми поставщиками. Например, при использовании поставщика Microsoft Jet 4.0 OLE DB свойство Recordset.RecordCount всегда возвращает значение $-$1. Лучше всего использовать код, который не зависит от наличия данных

в объекте Recordset, например Worksheet.CopyFromRecordset, применять свойства BOF и EOF или оператор For Each (оператор For Each обрабатывает только те записи, которые существуют в объекте Recordset).

Свойство Filter

Свойство Filter позволяет фильтровать содержимое открытого набора записей. В результате фильтрации будут видимы только те записи, которые соответствуют критерию фильтрации. Данное свойство выполняет функцию дополнительного предиката WHERE над набором записей. Невидимые записи не удаляются и не меняются, но они становятся недоступны для операций над набором записей. Этому свойству можно присвоить строку, в которой описывается условие выбора записей, или одну из констант FilterGroupEnum.

Можно установить несколько фильтров. В таком случае будут доступны только те записи, которые удовлетворяют всем условиям фильтров. Для удаления фильтров из набора записей установите свойство Filter равным пустой строке или константе adFilterNone. В следующем фрагменте кода демонстрируется фильтрация набора записей для отображения регионов, названия которых содержат подстроку "OR", например, "Oregon":

```
Recordset.Filter = "Region = 'OR'"
```

Для установки дополнительных фильтров можно воспользоваться логическими операторами AND, OR или NOT:

```
Recordset.Filter = "Region = 'OR' AND City = 'Portland'"
```

Свойство State

Свойство Recordset.State имеет тот же набор возможных значений, что и свойство Connection.State. (Дополнительная информация доступна в разделе "Свойство State" ранее в этой главе.)

Методы объекта Recordset

Классы могут быть большими и сложными, как класс Recordset. Но знание основных свойств и методов, а также понимание основных принципов их применения позволяет сразу эффективно использовать этот класс и изучать дополнительные возможности по мере необходимости. В этом разделе рассматривается пять часто используемых методов класса Recordset.

Метод Open

Метод Open извлекает данные и делает их доступными для кода. Метод Open имеет следующий синтаксис:

```
Call Recordset.Open(Source, ActiveConnection, CursorType,
↳ LockType, Options)
```

Аргумент Source описывает источник данных, он также может быть оператором SQL, объектом Command, именем таблицы, указателем URL, вызовом хранимой процедуры или сохраненным в файле набором записей. Обычно используется оператор SQL.

Аргумент ActiveConnection может содержать строку подключения или объект Connection, который идентифицирует применяемое подключение. Если в качестве значения аргумента ActiveConnection используется строка подключения, объект Connection создается автоматически. Аргумент CursorType описывает тип курсора, который используется при открытии набора записей. Тип курсора указывается с помо-

щью одного из значений CursorTypeEnum. В этой главе демонстрируется применение двух (adOpenForwardOnly и adOpenStatic) из четырех (adOpenForwardOnly, adOpenStatic, adOpenkeyset и adOpenDynamic) типов курсора. Знакомство с остальными типами курсора остается читателю в качестве самостоятельного упражнения.

Значение adOpenForwardOnly означает, что набор записей может просматриваться только в одном направлении, с начала в конец. Открытие набора записей с использованием adOpenForwardOnly является самым быстрым, но наименее гибким методом перемещения по набору записей. Кроме этого, однонаправленный набор данных поддерживает только чтение. Тип курсора adOpenStatic используется для отключенных наборов данных. Применение курсора adOpenStatic поддерживает произвольное перемещение и модификацию. Если тип курсора не указывать, по умолчанию используется adOpenForwardOnly.

Аргумент LockType указывает тип блокировки, которую поставщик применяет по отношению к источнику данных при создании набора записей. Существует пять типов блокировки: adLockBatchOptimistic, adLockOptimistic, adLockPessimistic, adLockReadOnly и adLockUnspecified. Далее в этой главе будут использоваться adLockReadOnly и adLockBatchOptimistic. Блокировка adLockBatchOptimistic применяется вместе с отключенными наборами записей, в которых обновление записей выполняется в пакетном режиме. Блокировка adLockOptimistic блокирует записи при вызове метода Update. При этом делается предположение, что никто не модифицировал записи с момента загрузки и до момента обновления. Блокировка adLockPessimistic блокирует записи сразу после начала модификации. Блокировка adLockReadOnly означает, что набор записей не может использоваться для модификации записей. Такая блокировка применяется только вместе с однонаправленными наборами записей. Блокировка adLockUnspecified означает неопределенную стратегию блокировки. Изучение механизмов блокировки, не рассмотренных в примерах данной главы, остается в качестве самостоятельного упражнения для читателей.

Аргумент Options совпадает с аргументом Options метода Execute объекта Connection, который рассматривался ранее в этой главе. Этот аргумент используется для отправки поставщику указаний по интерпретации содержимого аргумента Source.

Метод Close

Метод Close закрывает объект Recordset. При этом выделенная для хранения набора записей память не освобождается. Для ее освобождения необходимо присвоить объектной переменной Recordset значение Nothing. Так как язык VBA пытается быть дружественным к разработчику, объект удаляется из памяти, как только выходит за пределы области видимости. Например, если определить набор записей в методе, набор записей будет удален из памяти после завершения работы метода. С другой стороны, в некоторых языках необходимо явное освобождение памяти, выделенной для хранения объектов, поэтому лучше сформировать привычку присваивания объектным переменным значения Nothing (явного освобождения памяти).

Методы перемещения курсора

При первом открытии набора записей *указатель на текущую запись (current record pointer)* указывает на первую запись в наборе. Для перемещения по записям используется метод Move. Для этого метод перемещает текущий указатель на запись объекта Recordset. Ниже показаны базовые методы перемещения, доступные разработчику.

- MoveFirst — перемещает указатель на первую запись набора.
- MovePrevious — перемещает указатель на предыдущую запись.
- MoveNext — перемещает указатель на следующую запись.
- MoveLast — перемещает указатель на последнею запись набора.

В следующем примере показано перемещение между записями набора записей:

```
Public Sub RecordsetNavigation()

    Const SQL As String = _
        "SELECT * FROM Customers"
    Const ConnectionString As String = _
        "Provider=Microsoft.Jet.OLEDB.4.0;" + _
        "Data Source=C:\Program Files\Microsoft " + _
        "Office\OFFICE11\SAMPLES\Northwind.mdb;Persist Security" + _
        "Info=False"

    Dim Recordset As Recordset
    Set Recordset = New Recordset
    Call Recordset.Open(SQL, ConnectionString)

    Recordset.MoveFirst

    While Not Recordset.EOF
        Debug.Print Recordset.Fields("CompanyName")
        Recordset.MoveNext
    Wend

End Sub
```

Обратите особое внимание на использование метода MoveNext внутри цикла While. Неправильная запись такой конструкции является распространенной ошибкой, которая ведет к появлению бесконечного цикла.

Если необходимо изменить порядок записей в наборе, можно воспользоваться предложением ORDER BY. Использование предложения ORDER BY CompanyName DESC позволит получить отсортированные в порядке убывания данные. Кроме этого, в качестве типа курсора можно указать константу adOpenDynamic, переместиться на последнюю запись и перемещаться обратно в сторону первой записи. Модифицированный вариант кода показан ниже:

```
Public Sub RecordsetNavigation()

    Const SQL As String = _
        "SELECT * FROM Customers"

    Const ConnectionString As String = _
        "Provider=Microsoft.Jet.OLEDB.4.0;" + _
        "Data Source=C:\Program Files\Microsoft " + _
        "Office\OFFICE11\SAMPLES\Northwind.mdb;Persist Security" + _
        "Info=False"

    Dim Recordset As Recordset
    Set Recordset = New Recordset
    Call Recordset.Open(SQL, ConnectionString, adOpenDynamic)

    Recordset.MoveLast

    While Not Recordset.BOF
```

```
        Debug.Print Recordset.Fields("CompanyName")
        Recordset.MovePrevious
    Wend

End Sub
```

Метод NextRecordset

Некоторые поставщики поддерживают команды, которые возвращают несколько наборов записей. Метод NextRecordset используется для переключения между этими наборами. Вызов данного метода приводит к удалению текущего набора записей из объекта Recordset и загрузке нового набора записей. Текущий указатель устанавливается на первую запись в новом наборе записей. Если в момент вызова метода NextRecordset других наборов записей не осталось, объекту Recordset присваивается значение Nothing.

События объекта Recordset

Для перехвата событий объекта Recordset в модуле класса необходимо определить объектную переменную с помощью оператора WithEvents Recordset. Перехват событий необходим при асинхронном применении объекта Recordset, так как события используются для уведомления кода о завершении асинхронного выполнения задачи. Асинхронное поведение требует механизма уведомления о завершении. Часто вместе с асинхронными операциями используются события и механизмы блокирования. Как и в случае с объектом Connection, разработчика должны интересовать события, возникающие при использовании асинхронных методов набора записей: FetchComplete и FetchProgress. Применение этих событий с помощью оператора WithEvents и генерация процедур обработки событий средствами редактора VBE уже рассматривались ранее. Изучение асинхронного поведения объекта Recordset остается читателям в качестве самостоятельного упражнения.

Коллекции объекта Recordset

В заключение рассмотрения объекта Recordset стоит уделить внимание объектам, которые отслеживаются объектом Recordset — это коллекции Fields и Properties. Коллекция есть коллекция. Если умеешь использовать одну коллекцию, то умеешь использовать и все. Коллекции различаются хранимыми объектами. Таким образом, основное внимание будет уделяться классам объектов, которые хранятся в коллекциях Fields и Properties.

Коллекция Fields

В коллекции Fields хранятся таблицы, представления и наборы записей. Объект Field является пересечением строки и столбца в источнике данных. Коллекция Fields содержит все поля набора записей, ссылаясь на каждую строку по очереди. В коллекции Fields хранятся объекты класса Field, которые в основном используются для хранения значения единственного пересечения строки и столбца. Обычно интерес вызывает значение поля, но иногда возникает необходимость в информации об имени, типе, размере, индексе или ограничениях поля. В следующем примере показано, как отображать состояние поля. Эта информация дает представление о значениях записей и полей:

```
Public Sub DescribeARow()

    Const SQL As String = _
        "SELECT * FROM Customers"
```

```
    Const ConnectionString As String = _
        "Provider=Microsoft.Jet.OLEDB.4.0;" + _
        "Data Source=C:\Program Files\Microsoft " + _
        "Office\OFFICE11\SAMPLES\Northwind.mdb;Persist Security" + _
        "Info=False"

    Dim Recordset As Recordset
    Set Recordset = New Recordset
    Call Recordset.Open(SQL, ConnectionString, adOpenDynamic)

    Recordset.MoveFirst

    Dim Field As Field

    For Each Field In Recordset.Fields

        Debug.Print "Имя: " & Field.Name
        Debug.Print "Тип: " & Field.Type
        Debug.Print "Размер: " & Field.ActualSize
        Debug.Print "Значение: " & Field.Value
        Debug.Print "************************"
    Next

End Sub
```

В первой половине метода инициализируется и заполняется объект Recordset. Цикл For Each перебирает каждый столбец в наборе записей и выдает информацию о каждом поле в окне Immediate (Проверка).

Обычно желательно иметь представление обо всех аспектах изучаемой среды, но, скорее всего, из конкретных свойств поля придется использовать только имя, индекс и значение. Однако остальные свойства поля могут оказаться полезными на других уровнях. Самой распространенной причиной чтения атрибутов поля является создание динамических приложений. Например, информация об имени, типе, размере и значении поля теоретически позволяет на лету создавать графический пользовательский интерфейс. В языке VBA эта возможность реализована не в полной мере, но в таких средах, как Delphi и Visual Basic .NET предоставляются необходимые средства для динамического создания пользовательского интерфейса.

Коллекция Properties

В коллекции Properties хранятся специфичные для поставщиков и расширенные свойства объекта Recordset. Некоторые поставщики предоставляют расширенные свойства, о которых разработчику стоит знать. Расширенных свойств еще больше, чем производителей баз данных. Важно обратить внимание, что свойства имеют форму пар "имя-значение"; таким образом, используется одинаковый механизм чтения расширенных свойств. Коллекция Properties поддерживает индексацию по имени и номеру.

Объект Command

Объект Command является объектно-ориентированным представлением инструкций для поставщика. Одним из первых языков программирования (кроме RomBasic в молодости), с которым автору пришлось работать, был Databus. С созданием кода Databus его познакомил Майк Груp (Mike Groer). Databus является структурным языком, напоминающим COBOL. Язык зависит от состояния и поведения, но не поддерживает сущности

со связанным поведением. К сожалению, состояние и поведение не в полной мере отражают реальный мир, так как в реальном мире состояние и поведение связываются с определенными сущностями. Например, все люди дышат, но только некоторые являются пилотами. С точки зрения проектирования, все люди способны к обучению, а выбранные области обучения представляют состояние (или знания) каждого конкретного человека.

В результате эволюционного развития многие сущности в программировании приняли форму классов. Это относится и к командам SQL. Строка, в которой содержится команда SQL, является простой строкой, которая может содержать любое другое значение. Но строка внутри объекта `Command` должна содержать действительный оператор SQL (а проверка этой действительности является одним из аспектов поведения объекта). В этом разделе объект `Command` рассматривается более подробно.

Свойства объекта Command

В программировании все субъективно. Одной из любимых аналогий автора является представление кота в книге Гради Буча (Grady Booch). С точки зрения ветеринара кот имеет определенный набор генов и характеристик и состоит из определенных биологических атрибутов. С точки зрения любящей котов бабушки он является мурчащим другом. (С точки зрения брата автора коты являются отличными мишенями для тренировки с духовым ружьем. Шутка!) Таким образом, с точки зрения программиста состояние и поведение кота (как и любого другого класса) рассматривается с позиции предметной области, для которой создается решение. Для ветеринара подходит характеристика кота с биологической точки зрения. Для продавцов домашних животных достаточно будет указать цвет, характер, пол, размер и пройденные медицинские процедуры. (Если решение создается для брата, то придется указывать скорость, скрытность и успешность в охоте. Опять шутка!) Возвращаясь к классу `Command`, также можно отметить определенное развитие. Двадцать лет назад все источники данных представляли собой монолитные плоские текстовые файлы. Десять лет назад появилось упоминание логических отношений. В настоящий момент в концепцию источника данных входит все, включая файлы XML. Следовательно, в данный момент объект `Command` может содержать имена таблиц, запросы SQL и имена хранимых процедур. Ниже рассматриваются состояния и возможности объекта `Command`, которые доступны в настоящий момент.

Свойство ActiveConnection

Свойство `ActiveConnection`, уже рассматриваемое ранее, указывает на объект подключения, созданный явно или неявно с помощью строки подключения. (Использование свойства `ActiveConnection` демонстрируется в примерах кода ранее в этой главе.)

Свойство CommandText

Содержимое свойства `CommandText` (строковое значение) используется поставщиком данных для определения множества информации, которая будет предоставляться в составе набора данных.

Свойство CommandType

Свойство `CommandType` содержит подсказку для поставщика, помогающую правильно интерпретировать содержимого свойства `CommandText`. Если в свойстве `CommandText` передается имя хранимой процедуры, свойство `CommandType` должно иметь значение `CommandTypeEnum.adStoredProcedure`. Если в свойстве `CommandText` передается имя файла с набором записей, свойство `CommandType` должно быть установлено

в значение CommandTypeEnum.adCmdFile. Дополнительная информация доступна в справочном руководстве в разделах, посвященных свойствам Command.CommandText и Command.CommandType.

Методы объекта Command

Объект Command предоставляет только три метода. (Помните, что классы оцениваются с точки зрения простоты использования, полезности и эффективности. Если класс предоставляет только три метода, это не значит, что класс не важен или бесполезен.) Ниже рассматривается применение методов CreateParameter, Cancel и Execute.

Метод CreateParameter

Этот метод используется для самостоятельного создания объектов Parameter, которые могут быть добавлены в коллекцию Command.Parameters. Метод CreateParameter имеет следующий синтаксис:

```
CreateParameter([Name As String], _
 [Type As DataTypeEnum = adEmpty], _
 [Direction As ParameterDirectionEnum = adParamInput], _
 [Size As ADO_LONGPTR], [Value]) As Parameter
```

Name — это имя аргумента, которое можно использовать для обращения к объекту Parameter в коллекции Parameters объекта Command. При работе с SQL Server имя объекта Parameter должно совпадать с именем параметра хранимой процедуры, которому соответствует этот объект.

Параметр Type описывает тип данных параметра. Этот параметр может принимать значения, описанные в перечислимом типе DataTypeEnum. Каждое значение соответствует типу, который можно передавать в хранимую процедуру. Дополнительная информация о возможных значениях параметра приводится в справочном руководстве в разделе, посвященном типу DataTypeEnum.

Параметр Direction может иметь одно из значений типа ParameterDirectionEnum. Значение этого аргумента определяет предназначение параметра: параметр может использоваться для передачи данных входного аргумента, получения данных выходного аргумента или для получения возвращаемого значения хранимой процедуры. Имена констант, которые можно присваивать этому аргументу, говорят сами за себя: adParamInput, adParamInputOutput, adParamOutput и adParamReturnValue.

Аргумент Size используется для указания размера значения в байтах. Аргумент Value содержит значение аргумента, передаваемого в команду.

В следующем примере показано применение метода CreateParameter совместно с методом Append коллекции Parameters для создания объекта Parameter и добавления его в коллекцию Parameters с помощью единственной строки кода. Далее в примере выполняется подключение к серверу SQL Server, который хранит базу данных Northwind, и вызывается хранимая процедура Sales by Year:

```
Public Sub CallStoredProcedure()

   Const ConnectionString As String = _
     "Provider=SQLOLEDB.1;Integrated Security=SSPI;" + _
     "Persist Security Info=False;Initial Catalog=NorthwindCS;" + _
        "Data Source=LAP800;Workstation ID=LAP800;"


  Dim Command As Command
```

```
10:     Set Command = New Command
11:
12:   Command.ActiveConnection = ConnectionString
13:   Command.CommandText = "[Sales by Year]"
14:   Command.CommandType = CommandTypeEnum.adCmdStoredProc
15:
16:   Dim BeginningDate As ADODB.Parameter
17:   Dim EndingDate As ADODB.Parameter
18:
19:   Dim StartDate As Date
20:   StartDate = #1/1/1995#
21:
22:   Dim EndDate As Date
23:   EndDate = #1/1/2004#
24:
25:   Set BeginningDate = Command.CreateParameter("@Beginning_Date", _
26:   DataTypeEnum.adDate, ParameterDirectionEnum.adParamInput, ,
↳        StartDate)
27:
28:     Set EndingDate = Command.CreateParameter("@Ending_Date", _
29:         DataTypeEnum.adDate, ParameterDirectionEnum.adParamInput,
↳ , EndDate)
30:
31:     Call Command.Parameters.Append(BeginningDate)
32:     Call Command.Parameters.Append(EndingDate)
33:
34:     Dim Recordset As ADODB.Recordset
35:     Set Recordset = Command.Execute
36:
37:     Call Sheet1.Range("A1").CopyFromRecordset(Recordset)
38:
39:End Sub
```

Новая строка подключения определяется в строках с 3 по 6. В строках 9 и 10 определяется и инициализируется объект Command. В строке 12 объект Command связывается со строкой подключения. При этом объект Command самостоятельно создает объект Connection. С другой стороны, можно создать объект Connection и присвоить его свойству ActiveConnection. В строке 13 указывается имя хранимой процедуры, а в строке 14 свойство CommandType устанавливается в значение CommandTypeEnum.adCmdStoredProc. На основе этого значения объект Command принимает решение об интерпретации содержимого свойства CommandText. В строках 16 и 17 определяются два параметра, которые передаются хранимой процедуре Sales by Year. В строках с 19 по 32 определяются и инициализируются два значения, используемые для инициализации параметров. Обычно такой подробный код не требуется, но он проще в отладке и легче для понимания.

В строках 25, 26, 28 и 29 метод Command.CreateParameter применяется для инициализации каждого параметра. Имена параметров можно получить из хранимой процедуры. Тип параметра определяется точно так же. Так как два параметра используются в качестве аргументов, параметры инициализируются с помощью значения ParameterDirectionEnum.adParamInput и фактического значения параметра. Наконец, объекты параметров добавляются в коллекцию Parameters объекта Command (строки 31 и 32), запускается выполнение команды (строка 35) и копия полученного набора данных копируется на лист.

Метод Execute

Этот метод выполняет текст команды, который предоставлен в качестве значения свойства CommandText объекта Command. Следующий синтаксис метода Execute используется для запросов на выполнение операции (запросов, которые не приводят к возврату набора данных):

```
Call Command.Execute( [RecordsAffected], [Parameters], [Options])
```

Для запроса на возврат данных применяется следующий синтаксис:

```
Set Recordset = Command.Execute([RecordsAffected], [Parameters],
[Options])
```

Аргументы RecordsAffected и Options аналогичны соответствующим аргументам метода Execute объекта Connection. Этот метод описывался в разделе "Методы объекта Connection" ранее в этой главе. При выполнении оператора SQL, требующего передачи большего количества параметров, аргументу Parameters можно присвоить массив значений, по одному значению для каждого обязательного параметра. (Пример передачи массива параметров приводился в конце предыдущего раздела.)

Коллекции объекта Command

В примере использования метода CreateParameter были показаны все составляющие, необходимые для запуска команды. В конце обсуждения рассмотрим коллекции Parameters и Properties.

Коллекция Parameters

Коллекция Parameters содержит все объекты Parameter, связанные с объектом Command. Параметры используются для передачи аргументов оператору SQL и хранимым процедурам, а также для получения вывода и возвращаемых значений хранимых процедур.

Коллекция Properties

Коллекция Properties содержит специфичные для поставщика или расширенные свойства объекта Command. Некоторые производители предоставляют важные настройки, о которых стоит знать разработчикам. Дополнительная информация о специфичных для поставщика свойствах доступна в справочном руководстве.

Использование ADO в приложениях Microsoft Excel

В этом разделе вся информация будет сведена воедино. Таким образом, речь пойдет об одновременном использовании показанных ранее принципов программирования в Excel и методов работы с библиотекой ADO и языком SQL. Приложения Excel часто применяются для работы с данными, которые хранятся во внешних источниках. Чаще всего в качестве источника данных выступает база данных Access или SQL Server, но существуют приложения, получающие доступ к данным из текстового файла и даже из книг Excel. Далее будет показано, что библиотека ADO значительно упрощает получение данных из разных источников.

> В следующих двух разделах используется база данных Northwind. Эта база данных предоставляется в качестве примера в составе Access и SQL Server. Если база данных не установлена, установите ее перед запуском примеров кода.

Для запуска примеров кода, выберите пункт Tools⇨References (Сервис⇨Ссылки) в редакторе VBE. Откроется диалоговое окно References (Ссылки). Прокрутите список и найдите запись Microsoft ActiveX Data Objects 2.7 Library. Установите флажок напротив этой записи (как показано на рис. 11.5) и щелкните на кнопке OK.

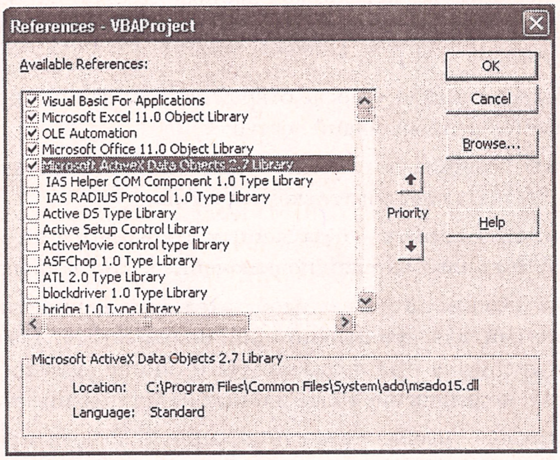


Рис. 11.5. Подключение библиотеки Microsoft ActiveX Data Objects 2.7 Library

Обратите внимание, что в системе может существовать несколько версий библиотеки ADO.

Использование библиотеки ADO вместе с Microsoft Access

Остаток этой главы посвящен примерам использования баз данных Microsoft Access и Microsoft SQL Server. Сначала рассмотрим применение баз данных Access, после чего — Microsoft SQL Server. (Важно помнить, что библиотека ADO поддерживает использование любого источника данных, а примеры в этой главе выбраны из-за доступности соответствующих баз данных.)

Подключение к Microsoft Access

Для подключения к базам данных Microsoft Access библиотека ADO использует поставщик OLE DB для Microsoft Jet (Jet — это механизм управления базами данных, применяемый в Access). В данном случае используется поставщик версии 4.0. Для подключения к базе данных Microsoft Access достаточно указать поставщика ADO в строке подключения и специфичные для поставщика параметры. Ранее в этой главе было показано, как создавать строку подключения с помощью диалогового окна Data Link Properties. Стоит немного подробнее рассмотреть составляющие строки подключения к базе данных Access:

- `Provider=Microsoft.Jet.OLEDB.4.0` — обязательный компонент строки подключения, описывающий поставщика OLE DB, который будет использоваться для установки подключения;

- Data Source=[полный путь и имя файла базы данных Access] — обязательный компонент строки подключения, который указывает путь к файлу базы данных;
- Mode=режим — необязательный компонент строки подключения, описывающий допустимые операции над источником данных. Доступные типы подключения перечислены в типе ConnectModeEnum. Чаще всего используются следующие значения параметра:
  - adModeShareDenyNone — после открытия базы данных другим пользователям предоставляется полный общий доступ;
  - adModeShareDenyWrite — после открытия базы данных другим пользователям предоставляется доступ только для чтения;
  - adModeShareExclusive — база данных открывается в однопользовательском режиме. Другим пользователям подключение к базе данных не предоставляется;
- User ID=имя_пользователя — необязательный компонент строки подключения, используемый для аутентификации пользователя. Если для подключения к базе данных требуется имя пользователя и оно не указано в качестве значения этого аргумента, подключение к базе данных не устанавливается;
- Password=пароль — необязательный компонент строки подключения. Этот аргумент используется в паре с User ID для аутентификации запроса на подключение. Если база данных защищена паролем, этот аргумент является обязательным.

В следующем примере показана строка подключения, в которой применяются все аргументы, описанные выше:

```
Public Const ConnectionString As String = _
    "Provider=Microsoft.Jet.OLEDB.4.0;" & _
    "Data Source=C:\Files\Northwind.mdb;" & _
    "Mode=Share Exclusive;" & _
    "User ID=Admin;" & _
    "Password=password"
```

Обратите внимание, что при инициализации объекта Connection с помощью полной строки подключения, для указания значения Mode используются удобочитаемые значения. Например, при определении полной строки подключения для аргумента Mode можно применять строку Share Exclusive. Но если инициализировать свойство Connection.Mode из кода, то придется использовать эквивалентное значение ConnectModeEnum.adModeShareExclusive. Самым надежным методом создания строки подключения для любого поставщика является применение диалогового окна Data Link Properties.

Получение данных из базы данных Microsoft Access с помощью простого запроса

В следующей процедуре показано получение данных из базы данных Microsoft Access с помощью простого текстового запроса. Полученные данные копируются на лист Excel.

```
Public Sub PlainTextQuery()

    Const ConnectionString As String = _
        "Provider=Microsoft.Jet.OLEDB.4.0;" + _
        "Data Source=C:\Program Files\Microsoft " + _
```

```
        "Office\OFFICE11\SAMPLES\Northwind.mdb;Persist Security" + _
        "Info=False"

    Dim Recordset As ADODB.Recordset

    ' Определение оператора SQL
    Const SQL As String = _
        "SELECT CompanyName, ContactName " & _
        "FROM Customers " & _
        "WHERE Country = 'UK' " & _
        "ORDER BY CompanyName"

    ' Инициализация объекта Recordset и отправка запроса
    Set Recordset = New ADODB.Recordset
    Call Recordset.Open(SQL, ConnectionString, _
         CursorTypeEnum.adOpenForwardOnly, _
        LockTypeEnum.adLockReadOnly, CommandTypeEnum.adCmdText)

    ' Удостоверится, что записи получены
    If Not Recordset.EOF Then
        ' Копирование полученных записей на лист
        Call Sheet1.Range("A2").CopyFromRecordset(Recordset)

        ' Добавить заголовки
        With Sheet1.Range("A1:B1")
            .Value = Array("Название компании",
↳"Имя контактного лица")
            .Font.Bold = True
        End With

        ' Установить ширину столбца в соответствии с размером данных
        Sheet1.UsedRange.EntireColumn.AutoFit
    Else
        Call MsgBox("Ошибка: не получено ни одной записи.", vbCritical)
    End If

    ' Удалить объект Recordset, если он еще существует
    If (Recordset.State And ObjectStateEnum.adStateOpen) Then
↳ Recordset.Close
    Set Recordset = Nothing

End Sub
```

В данном случае из библиотеки ADO применяется только объект Recordset. Как было показано в начале раздела, посвященного библиотеке ADO, все объекты верхнего уровня могут создаваться и использоваться независимо. Если в процессе работы приложения планируется выполнять несколько запросов, стоит создать отдельный общедоступный объект Connection. Это позволит использовать возможность библиотеки ADO создавать пул подключений.

Синтаксис метода Recordset.Open оптимизирован для получения максимальной производительности. Поставщику сообщается тип команды, которая передается в аргументе Source (adCmdText, простой текстовый запрос), и используется однонаправленный курсор, предназначенный только для чтения. Значение свойства CursorLocation не указывается. Часто такой курсор называют *брандсбойтом (firehose)*, так как это самый быстрый способ получения данных из базы данных.

Модификации в интересующий лист не вносятся до успешного получения данных из базы данных. В таком случае не придется отменять изменения, если обработка запроса

завершится неудачно. После получения данных метод CopyFromRecordset позволяет переместить данные из объекта Recordset на лист электронной таблицы. В конце процедуры выполняется форматирование заголовков столбцов и удаление объекта Recordset.

Получение данных из Microsoft Access с помощью хранимого запроса

База данных Microsoft Access поддерживает создание и сохранение запросов SQL в базе данных. Сохраненные запросы позволяют получать данные так же, как и при использовании простых текстовых запросов. В следующей процедуре показано применение запроса, сохраненного в базе данных Access:

```
Public Sub SavedQuery()

    Dim Field As ADODB.Field
    Dim Recordset As ADODB.Recordset
    Dim Offset As Long
        Const ConnectionString As String = _
            "Provider=Microsoft.Jet.OLEDB.4.0;" + _
            "Data Source=C:\Program Files\Microsoft " + _
            "Office\OFFICE11\SAMPLES\Northwind.mdb;
↳Persist Security Info=False"

    ' Создание объекта Recordset и отправка запроса
    Set Recordset = New ADODB.Recordset
    Call Recordset.Open("[Sales By Category]", ConnectionString, _
        CursorTypeEnum.adOpenForwardOnly, _
        LockTypeEnum.adLockReadOnly, _
        CommandTypeEnum.adCmdTable)

    ' Проверка существования записей
    If Not Recordset.EOF Then
        ' Добавление заголовков на лист
        With Sheet1.Range("A1")
            For Each Field In Recordset.Fields
                .Offset(0, Offset).Value = Field.Name
                Offset = Offset + 1
            Next Field
            .Resize(1, Recordset.Fields.Count).Font.Bold = True
        End With

        ' Копирование набора записей на лист
        Call Sheet1.Range("A2").CopyFromRecordset(Recordset)

        ' Изменение ширины столбцов в соответствии с размером данных
        Sheet1.UsedRange.EntireColumn.AutoFit
    Else
        MsgBox "Ошибка: не получено ни одной записи.", vbCritical
    End If

    ' Удаление объекта Recordset
    Recordset.Close
    Set Recordset = Nothing

End Sub
```

Обратите внимание на различия между вызовом Recordset.Open, который используется в этой процедуре, и вызовом с применением простого текстового запроса. В этом случае, вместо предоставления строки с оператором SQL, указывается имя сохраненного запроса, который должен использоваться для получения данных. Кроме этого, постав-

щику сообщается тип запроса (табличный запрос). Поставщик Jet OLE DB считает сохраненные запросы и запросы всего содержимого таблиц базы данных одинаковыми.

Так как оператор SQL создавался другими разработчиками, имена получаемых полей заранее неизвестны. Также неизвестно количество получаемых полей. Таким образом, для создания правильного набора заголовков для столбцов на листе необходимо просмотреть коллекцию `Fields` объекта `Recordset` и получить эту информацию динамически. Для этого объект `Recordset` должен оставаться открытым, поэтому поля на листе добавляются до удаления объекта `Recordset`.

Вставка, обновление и удаление записей в базе данных Microsoft Access с помощью простого текстового запроса SQL

Для выполнения текстовых запросов INSERT, UPDATE и DELETE используется одинаковый подготовительный код. Таким образом, добавление, обновление и удаление записи будет показано в пределах одной процедуры. Обычно так не делается, но данную универсальную процедуру можно превратить в специализированную простым удалением ненужного раздела.

В этом случае будет использоваться таблица `Shippers` из базы данных `Northwind`. Оригинальное содержимое таблицы показано на рис. 11.6. Перед модификацией этой таблицы стоит обратить внимание на два момента. Во-первых, заголовки таблицы немного отличаются от имен полей, которые используются в запросах SQL. Это связано с тем, что в базе данных Access с каждым полем таблицы связано свойство `Caption`. Значение свойства `Caption` отображается вместо имени поля. При этом в запросах SQL необходимо использовать имена полей. Во-вторых, обратите внимание, что в последней строке столбца `Shipper ID` содержится значение `"(Auto Number)"`. На самом деле это не значение, а признак того, что значения в столбце `Shipper ID` генерируются автоматически механизмом управления базой данных Jet. Этот столбец представляет собой основной ключ таблицы `Shipper`, и поле `AutoNumber` является распространенным методом генерации уникального значения основного ключа. Как будет показано в следующем примере, изменение значения в поле `AutoNumber` невозможно. Если необходимо запомнить ссылку на вставленную в таблицу новую запись, придется извлечь из таблицы значение поля `AutoNumber`, которое было присвоено этой записи. Такая операция показана в следующем примере:

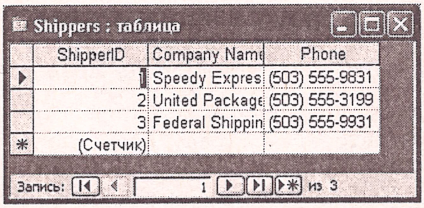


Рис. 11.6. Автоматически генерируемый первичный ключ

```
Public Sub CheckError(ByVal RecordsAffected As Long, _
       ByVal Expected As Long, ByVal Description As String)

   If RecordsAffected <> Expected Then
       Call RaiseError(Description)
```

```
    End If

End Sub

Public Sub RaiseError(ByVal Description As String)
        Call Err.Raise(vbObjectError + 1024, , Description)
End Sub


Public Function GetPrimaryKey(ByVal Command As ADODB.Command)
↳ As Long

    Dim RecordsAffected As Long
    Dim Recordset As ADODB.Recordset

    ' Извлечь основной ключ, сгенерированный для новой записи
    Command.CommandText = "SELECT @@IDENTITY"

    Set Recordset =
↳Command.Execute(Options:=CommandTypeEnum.adCmdText)

    If Recordset.EOF Then
        Call RaiseError("Ошибка получения основного ключа.")
    End If

    GetPrimaryKey = Recordset.Fields(0).Value
    Recordset.Close

    End Function

Public Sub ExecuteCommand(ByVal Command As ADODB.Command, _
        ByVal CommandText As String, _
        ByVal Description As String)

        Dim RecordsAffected As Long
        Command.CommandText = CommandText

    Call Command.Execute(RecordsAffected, , _
        CommandTypeEnum.adCmdText Or
↳ExecuteOptionEnum.adExecuteNoRecords)

    Call CheckError(RecordsAffected, 1, Description)

End Sub


Public Sub InsertRecord(ByVal Command As ADODB.Command)

        Const CommandText As String = _
        "INSERT INTO Shippers(CompanyName, Phone) " & _
        "VALUES('Air Carriers', '(205) 555 1212')"

    Const Description As String = _
        "Ошибка выполнения оператора INSERT."

    Call ExecuteCommand(Command, CommandText, Description)

End Sub

Public Sub UpdateRecord(ByVal Command As ADODB.Command,
↳ ByVal Key As Long)
```

```
        Dim CommandText As String
        CommandText = _
            "UPDATE Shippers SET Phone='(206) 546 0086' " & _
            "WHERE ShipperID=" & CStr(Key) & ";"

    Const Description As String = _
        "Ошибка выполнения оператора UPDATE."

    Call ExecuteCommand(Command, CommandText, Description)

End Sub

Public Sub DeleteRecord(ByVal Command As ADODB.Command, ByVal Key As Long)

    Dim CommandText As String
    CommandText = "DELETE FROM Shippers WHERE ShipperID = " & CStr(Key) & ";"

    Const Description As String = _
        "Ошибка выполнения оператора DELETE."

        Call ExecuteCommand(Command, CommandText, Description)

End Sub

Private Property Get ConnectionString() As String
    ConnectionString = _
        "Provider=Microsoft.Jet.OLEDB.4.0;" + _
        "Data Source=C:\Program Files\Microsoft " + _
        "Office\OFFICE11\SAMPLES\Northwind.mdb;Persist Security Info=False"

End Property


Public Sub InsertUpdateDelete()

    Dim Command As ADODB.Command
    Dim Key As Long

    On Error GoTo ErrorHandler

    Set Command = New ADODB.Command
    Command.ActiveConnection = ConnectionString

    Call InsertRecord(Command)
    Key = GetPrimaryKey(Command)

    Call UpdateRecord(Command, Key)

    Call DeleteRecord(Command, Key)

    ErrorExit:
        Set Command = Nothing
        Exit Sub

ErrorHandler:
    Call MsgBox(Err.Description, vbCritical)
    Resume ErrorExit
End Sub
```

Существует много методов декомпозиции решения. Предпочтительней использовать небольшие самодокументированные простые для понимания методы, которые просто отлаживать и можно применять повторно. Этот стиль демонстрируется в показанном выше листинге. Некоторые программисты (и авторы этой книги) могут создавать большие методы с большим количеством комментариев, но такие монолитные методы обычно все равно выполняют определенную задачу. В данном примере проблема разбита на такие составляющие: `CheckError` (проверка ошибок), `RaiseError` (выдача сообщений об ошибках), `GetPrimaryKey` (получение основного ключа), `ExecuteCommand` (выполнение команды), `InsertRecord` (вставка записи), `UpdateRecord` (обновление записи), `DeleteRecord` (удаление записи) и `InsertUpdateDelete`.

Процедуры `CheckError` и `RaiseError` используются для проверки количества измененных записей и выдачи сообщения об ошибке, если количество записей не совпадает с прогнозируемым. Процедуры `GetPrimaryKey`, `InsertRecord`, `UpdateRecord` и `DeleteRecord` используют процедуру `ExecuteCommand` для отправки запроса к базе данных и проверки результата. Процедура `GetPrimaryKey` вызывается сразу после команды `INSERT` (строки 107 и 108) и отправляет базе данных запрос `SELECT⇨IDENTITY`. В результате этого запроса выдается основной ключ, сгенерированный при выполнении последнего запроса `INSERT`. В процедурах `UpdateRecord` и `DeleteRecord` демонстрируется использование основного ключа для поиска конкретных записей. Наконец, в методе `InsertUpdateDelete` выполняются различные запросы. В этом методе демонстрируется важность повторного применения и обработки ошибок (строки с 114 по 121).

> Обратите внимание, что запрос `⇨IDENTITY` работает только в базах данных Access, которые сохранены в формате Access 2000 или более поздней версии. В Access 97 такие запросы не поддерживаются.

Использование ADO вместе с Microsoft SQL Server

В предыдущем разделе рассматривалось выполнение базовых запросов средствами библиотеки ADO. Так как библиотека ADO проектировалась для предоставления общего интерфейса для различных источников данных, базовые операции для Access ничем не отличаются от базовых операций для SQL Server. Следовательно, после краткого описания важных отличий применения ADO вместе с SQL Server, в этом разделе рассматриваются более важные темы, включая хранимые процедуры, множественные наборы записей и отключенные наборы записей. Использование SQL Server подробно не рассматривается, так как этот вопрос выходит за пределы тематики данной книги. Для получения дополнительной информации об использовании SQL Server стоит обратиться к одной из лучших книг: *Professional SQL Server 2000* Роберта Виера (Robert Vieira) от издательства Wrox (ISBN 1-861004-48-6).

Подключение к Microsoft SQL Server

Для подключения к базе данных Microsoft SQL Server в строке подключения ADO достаточно указать поставщика OLE DB для SQL Server и все необходимые параметры поставщика. Ниже приводится список часто используемых аргументов строки подключения, которые могут потребоваться при подключении к базе данных SQL Server.

- `Provider=SQLOLEDB`.
- `Data 2Source=имя_сервера` — если это SQL Server 7.0, то в качестве имени сервера указывается NetBIOS-имя компьютера, на котором установлен SQL Server. После появления SQL Server 2000 появилась возможность установки нескольких серверов на одном компьютере. В этом случае имя сервера будет выглядеть, как Имя NetBIOS\Имя SQL Server. Если SQL Server установлен на том же компьютере, на котором установлено приложение электронной таблицы, можно использовать имя `localhost`.
- `Initial Catalog=имя_базы_данных` — в отличие от Access один экземпляр сервера SQL Server может хранить несколько баз данных. Этот аргумент описывает имя базы данных, к которой необходимо подключиться.
- `User ID = имя_пользователя` — это имя используется при аутентификации на сервере SQL Server.
- `Password=пароль` — используется при аутентификации на сервере SQL Server.
- `NetworkLibrary=netlib` — по умолчанию поставщик OLE DB для SQL Server попытается подключиться к SQL Server с помощью *сетевого протокола именованных конвейеров (named pipes network protocol)*. Этот протокол необходим для получения доступа к внутренним механизмам безопасности операционной системы Windows (эти механизмы рассматриваются далее). Но иногда бывают ситуации, когда именованные конвейеры не работают. Такая проблема возникает при доступе к SQL Server с компьютера под управлением операционной системы Windows 9x, а также при доступе к серверу по сети Internet. В таком случае предпочтение отдается протоколу TCP/IP. Предпочтительный протокол может быть указан на каждом компьютере. Для этого следует воспользоваться утилитой SQL Server Network Utility или аргументом строки подключения NetworkLibrary для указания имени сетевой библиотеки TCP/IP, которая называется `dbmssocn`.
- `IntegratedSecurity=SSPI` — этот аргумент строки подключения указывает необходимость использования встроенных механизмов безопасности операционной системы Windows. Встроенные механизмы безопасности применяются вместо механизмов аутентификации SQL Server. При использовании этого аргумента, аргументы `UserID` и `Password` игнорируются.

Замечание об использовании механизмов безопасности SQL Server

Существует три механизма безопасности, поддерживаемых SQL Server: встроенная аутентификация SQL Server, механизмы безопасности операционной системы Windows и смешанный режим. Механизм аутентификации SQL Server требует добавления отдельных записей пользователей в SQL Server, также каждый пользователь перед подключением должен назвать имя и ввести пароль.

Такой механизм обеспечения безопасности чаще всего используется для предоставления доступа к SQL Server из-за пределов локальной сети. При применении интегрированных механизмов безопасности операционной системы Windows программное обеспечение SQL Server использует те же имена и пароли, что и для регистрации пользователей в сети Windows. Смешанный режим поддерживает применение любого метода аутентификации.

Ниже показан пример строки подключения, в которой присутствуют базовые элементы. Помните, диалоговое окно Data Link Properties поддерживает создание строки подключения не только к SQL Server, но и к множеству других поставщиков.

```
Const ConnectionString As String = _
    "Provider=SQLOLEDB.1;Integrated Security=SSPI;" + _
    "Persist Security Info=False;Initial Catalog=NorthwindCS;" + _
    "Data Source=LAP800;Workstation ID=LAP800;"
```

Хранимые процедуры Microsoft SQL Server

Синтаксис простых текстовых запросов к SQL Server не отличается от синтаксиса запросов к Access. Отличается только строка подключения. Так как текстовые запросы уже рассматривались, обратим внимание на вызов хранимых процедур, которые находятся в базе данных SQL Server.

Хранимые процедуры являются доступными по имени предварительно откомпилированными операторами SQL. Они очень напоминают процедуры VBA, так как хранимые процедуры поддерживают передачу аргументов и возврат значений. Ниже приводится пример простой хранимой процедуры, которая может использоваться для выполнения запросов к таблице `Orders` и представлению `Order Subtotal`:

```
ALTER Procedure dbo.[Employee Sales by Country] @Beginning_Date
↳ datetime, @Ending_Date datetime AS SELECT Employees.Country,
↳ Employees.LastName,
   Employees.FirstName, Orders.ShippedDate, Orders.OrderID,
   "Order Subtotals".Subtotal AS SaleAmount FROM Employees
↳ INNER JOIN (Orders INNER JOIN "Order Subtotals"
  ON Orders.OrderID = "Order Subtotals".OrderID)
  ON Employees.EmployeeID = Orders.EmployeeID WHERE
↳ Orders.ShippedDate
   Between @Beginning_Date And @Ending_Date
```

Эта хранимая процедура принимает два аргумента: `@Beginning_Date` и `@Ending_Date`. Она возвращает набор записей, в котором хранятся значения полей, перечисленных в предложении SELECT. Возвращаются значения записей, входящих в таблицу `Orders` и представление `Order subtotals`, значение поля `ShippedDate` которых попадает в период от `Beginning_Date` до `Ending_Date`.

Библиотека ADO предоставляет простой способ вызова хранимых процедур через объект `Connection`. Она рассматривает все хранимые процедуры в текущей базе данных в качестве динамических методов объекта `Connection`. Хранимую процедуру можно вызывать как и любой другой метод объекта `Connection`. Параметры хранимой процедуры допускается передавать как аргументы метода объекта. Если хранимая процедура возвращает набор записей, в качестве последнего необязательного аргумента можно передать ссылку на объект `Recordset`.

Этот метод больше подходит для одноразовых процедур, так как для многократного вызова этот способ является не самым эффективным. Преимуществом этого способа является простота программирования. В следующем примере показан вызов показанной выше хранимой процедуры в качестве метода объекта `Connection`:

```
Public Sub ExecuteStoredProcAsMethod()

    Dim Connection As ADODB.Connection
    Dim Recordset As ADODB.Recordset
```

```
    Const ConnectionString As String = _
      "Provider=SQLOLEDB.1;Integrated Security=SSPI;" + _
      "Persist Security Info=False;Initial Catalog=NorthwindCS;" + _
      "Data Source=LAP800;Workstation ID=LAP800;"

On Error GoTo Cleanup
    Set Connection = New ADODB.Connection
    Set Recordset = New ADODB.Recordset

    Call Connection.Open(ConnectionString)

    Dim StartDate As Date, EndDate As Date
    StartDate = #1/1/1995#
    EndDate = #1/1/2004#
    Connection.Employee_Sales_by_Country StartDate,
↳ EndDate, Recordset

    Call Sheet1.Range("A1").CopyFromRecordset(Recordset)

    Sheet1.UsedRange.EntireColumn.AutoFit

Cleanup:
If (Err.Number <> 0) Then Debug.Print Err.Description
If (Connection.State = ObjectStateEnum.adStateOpen)
↳ Then Connection.Close
If (Recordset.State = ObjectStateEnum.adStateOpen)
↳ Then Recordset.Close

End Sub
```

В этой процедуре вызывается хранимая процедура Employee_Sales_by_Country. Значения #1/1/1995# и #1/1/2004# передаются в качестве аргументов @Beginning_Date и @Ending_Date. В результате вызова хранимой процедуры возвращается массив записей о продажах, сделанных сотрудниками с 1995 по 2004 год. Обратите внимание, что объект Connection необходимо создать до наполнения динамических методов, а объект Recordset инициализировать до передачи его в качестве аргумента хранимой процедуры.

Самым эффективным способом работы с многократно вызываемыми хранимыми процедурами является подготовка глобального объекта Command, представляющего хранимые процедуры для остального кода. Объект Connection будет храниться внутри свойства ActiveConnection объекта Command. Хранимая процедура будет храниться в свойстве CommandText объекта Command, а аргументы хранимой процедуры передаются в виде членов коллекции Parameters объекта Command.

После создания объекта Command его можно вызывать несколько раз, не опасаясь накладных расходов, возникающих при выполнении описанных выше операций.

Например, создадим простую хранимую процедуру для вставки записей в таблицу Shippers:

```
ALTER PROCEDURE InsertShippers
   @CompanyName nvarchar(40),
   @Phone nvarchar(24)
AS
INSERT INTO Shippers
(CompanyName, Phone)
VALUES (@CompanyName, @Phone)
RETURN @@IDENTITY
```

```
CREATE PROCEDURE InsertShippers @CompanyName nvarchar(40),
↳ @Phone nvarchar(24) AS INSERT INTO Shippers
```

Не сложно заметить, что описанная хранимая процедура принимает два аргумента @CompanyName и @Phone, которые используются для вставки значений в соответствующие поля таблицы Shippers. В примере работы с базой данных Access было показано, что в таблице Shippers три поля. В хранимой процедуре первое поле, ShipperID, не обрабатывается.

Это связано с тем, что как и в таблице Shippers в базе данных Access Northwind, поле ShipperID в базе данных SQL Server Northwind заполняется автоматически. Новое значение поля генерируется при каждой операции вставки записи. Кроме этого, используется похожий способ извлечения автоматически сгенерированного значения. Для этого применяется системная функция SQL Server @@IDENTITY. Но в данном случае для получения значения поля ShipperID дополнительный вызов не требуется, так как значение сгенерированного поля возвращается в результате работы хранимой процедуры.

Для демонстрации реалистичного приложения в следующем примере используются глобальные объекты Connection и Command, процедуры для создания и удаления подключения, процедура для подготовки объекта Command к использованию, а также процедура, применяющая объект Command:

```
Option Explicit

Private Const ConnectionString As String = _
  "Provider=SQLOLEDB.1;Integrated Security=SSPI;" + _
  "Persist Security Info=False;Initial Catalog=NorthwindCS;" + _
  "Data Source=LAP800;Workstation ID=LAP800;"

Public Command As ADODB.Command
Public Connection As ADODB.Connection

Private Sub CreateConnection()
  Set Connection = New ADODB.Connection
  Call Connection.Open(ConnectionString)
End Sub

Private Sub DestroyConnection()
  If (Connection.State = ObjectStateEnum.adStateOpen) Then
    Connection.Close
  End If

  Set Connection = Nothing
End Sub

Private Sub PrepareCommandObject()

  Set Command = New ADODB.Command
  Set Command.ActiveConnection = Connection
  Command.CommandText = "InsertShippers"
  Command.CommandType = adCmdStoredProc

  Call Command.Parameters.Append( _
   Command.CreateParameter("@RETURN_VALUE", DataTypeEnum.adInteger, _
     ParameterDirectionEnum.adParamReturnValue, 0))

  Call Command.Parameters.Append( _
   Command.CreateParameter("@CompanyName", DataTypeEnum.adVarWChar, _
     ParameterDirectionEnum.adParamInput, 40))
```

```
  Call Command.Parameters.Append( _
    Command.CreateParameter("@Phone", DataTypeEnum.adVarWChar, _
      ParameterDirectionEnum.adParamInput, 24))

End Sub

Public Sub UseCommandObject()
  Dim Key As Long
  Dim RecordsAffected As Long

  On Error GoTo ErrorHandler

  CreateConnection
  PrepareCommandObject

  Command.Parameters("@CompanyName").Value = "Air Carriers"
  Command.Parameters("@Phone").Value = "(206) 555 1212"

  Call Command.Execute(RecordsAffected, , _
ExecuteOptionEnum.adExecuteNoRecords)
  If (RecordsAffected <> 1) Then
    Call Err.Raise(vbObjectError + 1024, , _
        Description:="Ошибка использования объекта Command.")
  End If

  Key = Command.Parameters("@RETURN_VALUE").Value
  Debug.Print "Значение ключа новой записи: " & CStr(Key)

ErrorExit:

    Set Command = Nothing
    DestroyConnection
    Exit Sub

ErrorHandler:
    Call MsgBox(Err.Description, vbCritical)
    Resume ErrorExit
End Sub
```

В обычном приложении объекты Connection и Command не создаются и не удаляются в процедуре UseCommandObject. Эти объекты предназначены для многократного использования, поэтому должны создаваться в начале работы приложения и удаляться перед завершением работы приложения.

При создании и использовании коллекции Parameters объекта Command не забывайте, что первый параметр всегда резервируется для возвращаемого значения хранимой процедуры, даже если она не имеет возвращаемого значения.

Несмотря на то что в этом примере возвращаемое значение сгенерированного поля ShipperID не используется, в реальных приложениях это значение играет очень важную роль. Поля CompanyName и Phone предназначены для человека. Механизм управления базой данных идентифицирует поле именно по значению основного ключа. Например, в базе данных Northwind поле ShipperID является обязательным для новых записей в таблице Orders. Таким образом, при добавлении заказа для нового поставщика потребуется новое значение поля ShipperID.

Несколько наборов записей

Поставщик SQL Server OLE DB поддерживает выполнение операторов SQL, которые возвращают несколько наборов записей. Эта возможность может показаться очень полезной при заполнении нескольких элементов управления диалогового окна на основе информации из базы данных. Все операторы поиска SELECT можно объединить в единственной хранимой процедуре. После этого перебирать наборы записей и вносить значения из каждого набора в соответствующие элементы управления диалогового окна.

Например, при создании пользовательского интерфейса для ввода информации в таблицу Orders потребуется информация из нескольких связанных таблиц, включая таблицы Customers и Shippers (см. рис. 11.7).

Создадим сокращенный пример хранимой процедуры для возврата информации из этих двух таблиц. Результат работы хранимой процедуры можно использовать для заполнения раскрывающихся списков в диалоговом окне UserForm:

```
CREATE PROCEDURE GetLookupValues
AS
  SELECT
     CustomerID,
     CompanyName
  FROM
    Customers
  SELECT
    ShipperID,
    CompanyName
  FROM
     Shippers
```

Обратите внимание, что в показанной выше хранимой процедуре используется два оператора SELECT. При вызове хранимой процедуры через интерфейс ADO эти операторы применяются для заполнения двух отдельных наборов записей. Ниже показано, как результат вызова хранимой процедуры GetLookupValues используется в процедуре обработки события aUserForm_Initialize для заполнения раскрывающихся списков в диалоговом окне.

```
Option Explicit

Private Const ConnectionString As String = _
    "Provider=SQLOLEDB.1;Integrated Security=SSPI;" + _
    "Persist Security Info=False;Initial Catalog=NorthwindCS;" + _
    "Data Source=LAP800;Workstation ID=LAP800;"

Private Sub UserForm_Initialize()

    Dim Connection As ADODB.Connection
    Dim Recordset As ADODB.Recordset

    Set Connection = New ADODB.Connection
    Connection.ConnectionString = ConnectionString

    Connection.Open

    Set Recordset = New ADODB.Recordset
    Call Recordset.Open("GetLookupValues", Connection, _
        CursorTypeEnum.adOpenForwardOnly, LockTypeEnum.adLockReadOnly, _
        CommandTypeEnum.adCmdStoredProc)
```

```
    Do While Not Recordset.EOF
        Call ComboBoxCustomers.AddItem(Recordset.Fields(1).Value)
        Recordset.MoveNext
    Loop

    Set Recordset = Recordset.NextRecordset

    Do While Not Recordset.EOF
        Call ComboBoxShippers.AddItem(Recordset.Fields(1).Value)
        Recordset.MoveNext
    Loop

    ' Неявно закрывает набор записей
    Set Recordset = Recordset.NextRecordset

    If (Connection.State = ObjectStateEnum.adStateOpen) Then
↪Connection.Close

End Sub
```

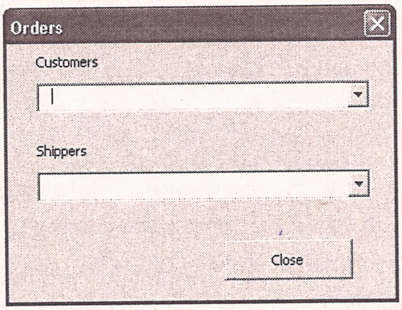


Рис. 11.7. Элементы управления, содержащие результат запроса к базе данных

Стоит отметить, что для использования этого метода необходимо заранее знать о количестве и порядке наборов записей, которые возвращаются в результате вызова хранимой процедуры. Кроме этого, здесь не обрабатываются значения основного ключа, связанные с описаниями таблицы поиска. В реальном приложении значения этих ключей пришлось бы сохранить (предпочтительным методом хранения можно считать объект `Collection`), что позволило бы извлекать значение основного ключа, соответствующее выбранному пользователем элементу раскрывающегося списка.

Отключенные наборы записей

В разделе "Получение данных из базы данных Microsoft Access с помощью простого запроса" было показано, что подключение и отключение от базы данных должно выполняться как можно быстрее. Но объект `Recordset` достаточно полезен, чтобы хранить его длительное время, не блокируя доступ к базе данных для других пользователей. Для решения этой проблемы можно воспользоваться возможностью библиотеки ADO, которая называется *отключенный набор записей (disconnected recordset)*.

Отключенный набор записей реализован в виде открытого объекта `Recordset` с разорванным подключением к источнику данных. В результате получается полностью функциональный объект `Recordset`, использование которого не требует блокировки базы данных. Отключенные наборы записей могут оставаться открытыми столько, сколько

требуется. Такие объекты синхронизируются с источником данных и даже сохраняются на диске для последующего извлечения. Некоторые из возможностей отключенного набора записей демонстрируются в следующем примере.

Предположим, что для пользователей необходимо реализовать возможность просмотра любой выбранной группы клиентов. Запрос к базе данных при каждом изменении критерия является крайне неэффективным подходом. Более подходящее решение — это запрос полного набора клиентов из базы данных и хранение этого набора в отключенном наборе записей. После этого можно воспользоваться свойством `Filter` объекта `Recordset` для быстрого извлечения запрошенного множества клиентов. В следующем примере показаны все компоненты, необходимые для создания отключенного набора записей:

```
Option Explicit

Private Const ConnectionString As String = _
    "Provider=SQLOLEDB.1;Integrated Security=SSPI;" + _
    "Persist Security Info=False;Initial Catalog=NorthwindCS;" + _
    "Data Source=LAP800;Workstation ID=LAP800;"

Public Connection As ADODB.Connection

Public Recordset As ADODB.Recordset

Public Sub CreateDisconnectedRecordset()

    Dim SQL As String
    SQL = "SELECT CustomerID, CompanyName, ContactName, Country " & _
        "FROM Customers"

        Set Connection = New ADODB.Connection
        Connection.ConnectionString = ConnectionString
        Connection.Open

        Set Recordset = New ADODB.Recordset
        Recordset.CursorLocation = CursorLocationEnum.adUseClient
        Recordset.CursorType = CursorTypeEnum.adOpenStatic

        Recordset.LockType = LockTypeEnum.adLockBatchOptimistic
        Call Recordset.Open(SQL, Connection, , ,
↳ CommandTypeEnum.adCmdText)

        Set Recordset.ActiveConnection = Nothing
        Call Sheet4.Range("A1").CopyFromRecordset(Recordset)

        ' Подключение намеренно остается открытым
End Sub
```

Обратите внимание, что объектная переменная типа `Recordset` объявлена глобальной. Если объявить переменную на уровне процедуры, интерпретатор VBA автоматически удалит переменную после завершения процедуры. В таком случае переменная окажется недоступной.

Существует шесть обязательных операций, которые необходимо выполнить для создания отключенного набора записей. Несколько операций можно скомбинировать в одну в момент вызова объекта `Recordset.Open`. Это решение более эффективно, но, для простоты, в данном случае эти операции разделены.

- Для начала необходимо создать пустой объект Recordset.
- Расположение курсора необходимо установить на стороне клиента. Так как набор записей будет отключен от сервера, управление курсором на сервере будет недоступно. Обратите внимание, что этот параметр необходимо установить до открытия набора записей. Изменение расположения курсора после открытия набора записей невозможно.
- Механизм управления курсором на стороне клиента, предоставляемый библиотекой ADO, поддерживает только один тип курсора — статический. Поэтому свойство CursorType необходимо установить в соответствующее значение.
- В библиотеке ADO существует специальный тип блокировки, предназначенный для отключенных наборов записей. Такая блокировка называется Batch Optimistic. Она позволяет подключать отключенный набор записей к базе данных и обновлять содержимое базы данных содержимым записей, модифицированных, пока набор записей был отключен. Эта операция выходит за пределы рассматриваемых в данной главе вопросов, поэтому просто обратите внимание, что для создания отключенного набора записей необходимо использовать блокировку Batch Optimistic.
- Следующим этапом является открытие набора записей. В этом примере используется простой текстовый запрос SQL. Это необязательно, так как отключенный набор записей можно создавать на основе любого источника данных, позволяющего создать стандартный набор записей. Существует несколько возможностей, которые отсутствуют в механизме управления курсором на стороне клиента, например, он не поддерживает несколько наборов записей одновременно.
- Завершающим этапом является отключение набора записей от источника данных. Для этого объекту Connection внутри объекта Recordset присваивается значение Nothing. В разделе "Свойства объекта Recordset" ранее в этой главе было показано, что связанный с объектом Recordset объект Connection доступен через свойство Recordset.ActiveConnection. Установка этого свойства в значение Nothing разрывает подключение между набором записей и источником данных.

Что можно делать с созданным отключенным набором данных? Ответ: практически все операции, которые поддерживаются объектом Recordset. Предположим, что пользователю необходим отсортированный в алфавитном порядке список клиентов, живущих в Германии. Для этого можно написать следующий код:

```
Public Sub FilterDisconnectedRecordset()
  Call Sheet4.Range("A:D").Clear
  Recordset.Filter = "Country = 'Germany'"
  Recordset.Sort = "CompanyName"
  Call Sheet4.Range("A1").CopyFromRecordset(Recordset)
End Sub
```

Если приходится работать в многопользовательской среде, данные в отключенном наборе данных устаревают в результате вставки, обновления и удаления записей другими пользователями. Для решения этой проблемы можно отправить повторный запрос на обновление содержимого объекта Recordset. Как показано в следующем примере, для этого достаточно повторно подключиться к источнику данных, вызвать метод Recordset.Requery и отключиться от источника данных:

```
Public Sub RequeryConnection()

    Set Recordset.ActiveConnection = Connection
    Call Recordset.Requery(Options:=CommandTypeEnum.adCmdText)
    Set Recordset.ActiveConnection = Nothing

End Sub
```

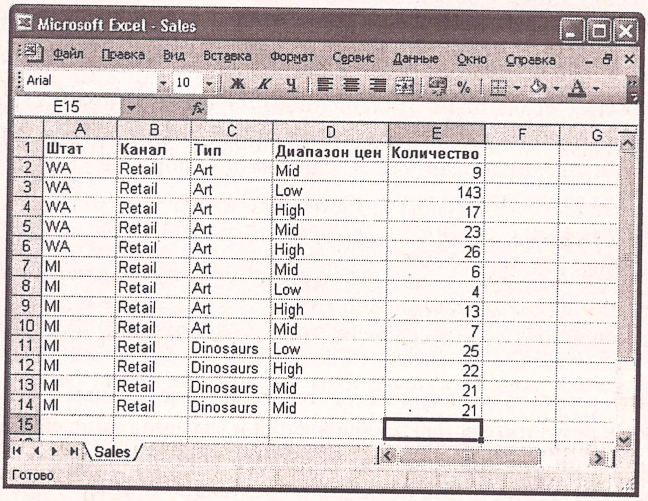
Использование библиотеки ADO для доступа к нестандартным источникам данных

В этом разделе описывается использование ADO для доступа к данным из двух распространенных нестандартных источников (эти источники данных нельзя назвать базами данных в строгом смысле) — книги Excel и текстовые файлы. Хотя смысл такого решения может показаться неочевидным, библиотека ADO часто оказывается лучшим инструментом для извлечения данных из книг и текстовых файлов, так как позволяет отказаться от длительной процедуры открытия этих файлов в Excel. Кроме этого, использование ADO позволяет применять всю мощь SQL для выполнения необходимых операций.

Запрос к книгам Microsoft Excel

При использовании ADO для доступа к книгам Excel применяется тот же поставщик OLE DB, который использовался ранее в этой главе для получения данных из базы данных Microsoft Access. Кроме Access, этот поставщик поддерживает большинство источников данных ISAM (ISAM data sources) (эти источники данных используют табличный формат на основе строк и столбцов). Чаще всего этот поставщик применяется для получения данных из закрытых книг Excel.

В данном случае будет использоваться книга `Sales.xls`, показанная на рис. 11.8. Эта книга, как и остальные примеры, доступна для загрузки на сайте Wrox.



	A	B	C	D	E	F	G
1	Штат	Канал	Тип	Диапазон цен	Количество		
2	WA	Retail	Art	Mid	9		
3	WA	Retail	Art	Low	143		
4	WA	Retail	Art	High	17		
5	WA	Retail	Art	Mid	23		
6	WA	Retail	Art	High	26		
7	MI	Retail	Art	Mid	6		
8	MI	Retail	Art	Low	4		
9	MI	Retail	Art	High	13		
10	MI	Retail	Art	Mid	7		
11	MI	Retail	Dinosaurs	Low	25		
12	MI	Retail	Dinosaurs	High	22		
13	MI	Retail	Dinosaurs	Mid	21		
14	MI	Retail	Dinosaurs	Mid	21		
15							

Рис. 11.8. Книга с примером базы данных

При применении ADO для доступа к данным в книгах Excel, книга используется в качестве базы данных, а листы и именованные диапазоны — в качестве таблиц. Сравним строку подключения к базе данных Access со строкой подключения к книге Excel:

Строка подключения к базе данных Access:

```
"Provider=Microsoft.Jet.OLEDB.4.0;" & _
            "Data Source=C:\Files\Northwind.mdb;"
```

Строка подключения к книге Excel:

```
"Provider=Microsoft.Jet.OLEDB.4.0;" & _
            "Data Source=C:\Files\Sales.xls;" & _
            "Extended Properties=Excel 8.0;"
```

Обратите внимание, что используется один и тот же поставщик. Вместо полного пути к базе данных Access применен полный путь к книге Excel. (В данном случае используются не настоящие имена файлов и пути. Замените путь и имя файла в соответствии с реальным расположением базы данных на компьютере.) Единственным отличием при применении поставщика OLE DB Microsoft Jet для подключения к другим источникам данных является необходимость указания имени источника данных в аргументе `Extended Properties`. При подключении к Excel 97 и более поздней версии параметр `Extended Properties` устанавливается в значение Excel 8.0.

Для запроса данных с листа Excel можно воспользоваться простым текстовым запросом SQL для получения данных из обычной базы данных, но при подключении к книге Excel формат имени таблицы отличается. Интересующую таблицу в книге Excel можно указать одним из четырех способов:

1. *Имя листа.* При указании имени листа в качестве имени таблицы в операторе SQL после имени листа необходимо добавить символ `$` и заключить имя в квадратные кавычки. Например, `[Sheet1$]` является допустимым именем таблицы при обращении к листу. Если имя листа содержит пробелы или неалфавитно-цифровые символы, его необходимо заключить в одинарные кавычки, например, `['My Sheet$']`.
2. *Имя диапазона на уровне листа.* Имя диапазона на уровне листа можно использовать в качестве имени таблицы в запросе SQL. Для этого достаточно перед именем диапазона указать имя листа, в котором находится этот диапазон, используя соглашения по форматированию, описанные выше, например, `[Sheet1$SheetLevelName]`.
3. *Адрес конкретного диапазона.* В качестве имени таблицы в запросе SQL можно использовать адрес конкретного диапазона на интересующем листе. Синтаксис этого метода идентичен синтаксису имени диапазона на уровне листа, например, `[Sheet1$A1:E20]`.
4. *Имя диапазона на уровне книги.* В запросе SQL в качестве имени таблицы можно использовать имя диапазона на уровне книги. В этом случае специальное форматирование не требуется. Имя используется без дополнительных квадратных кавычек.

Хотя в данной книге Excel содержится один лист, в ней может содержаться любое количество листов и именованных диапазонов. Главное, знать, к какому листу или диапазону обращаться в запросе. В следующей процедуре показано использование всех четырех методов обращения к таблицам:

```
Option Explicit

Public Sub QueryWorksheet()
```

```
    Dim Recordset As ADODB.Recordset
    Dim ConnectionString As String
        ConnectionString = _
        "Provider=Microsoft.Jet.OLEDB.4.0;" & _
        "Data Source=" & ThisWorkbook.Path & "\Sales.xls;" & _
        "Extended Properties=Excel 8.0;"

    Dim SQL As String

    ' Запрос на основе имени листа.
    SQL = "SELECT * FROM [Sales$]"
    ' Запрос на основе имени диапазона уровня листа.
    ' SQL = "SELECT * FROM [Sales$MyRange]"
    ' Запрос на основе конкретного имени диапазона.
    ' SQL = "SELECT * FROM [Sales$A1:E14]"
    ' Запрос на основе имени диапазона уровня книги.
    ' SQL = "SELECT * FROM BookLevelName"

    Set Recordset = New ADODB.Recordset

    On Error GoTo Cleanup

    Call Recordset.Open(SQL, ConnectionString, _
        CursorTypeEnum.adOpenForwardOnly,
↳LockTypeEnum.adLockReadOnly, _
        CommandTypeEnum.adCmdText)

    Call Sheet1.Range("A1").CopyFromRecordset(Recordset)

Cleanup:
    Debug.Print Err.Description

    If (Recordset.State = ObjectStateEnum.adStateOpen) Then
        Recordset.Close
    End If

    Set Recordset = Nothing

End Sub
```

По умолчанию поставщик OLE DB для Microsoft Jet предполагает, что первая строка указанной в запросе SQL таблицы содержит имена полей для данных. Если это так, то можно использовать более сложные запросы SQL с предложениями WHERE и ORDER BY. Если в первой строке таблицы данных не содержатся имена полей, поставщику необходимо об этом сообщить или первая строка с данными будет потеряна. Для этого необходимо установить дополнительный параметр HDR=No, который передается в составе аргумента Extended Properties в строке подключения:

```
"Provider=Microsoft.Jet.OLEDB.4.0;" & _
"Data Source=" & ThisWorkbook.Path & "\Sales.xls;" & _
"Extended Properties=""Excel 8.0;HDR=No"";"
```

Обратите внимание, что при передаче нескольких параметров в составе аргумента Extended Properties, всю строку аргумента необходимо заключить в двойные кавычки, а отдельные параметры — разделять точкой с запятой. Если в таблице с данными отсутствует строка с заголовками полей, поставщик поддерживает только простые запросы SELECT.

Вставка и обновление записей в книгах Microsoft Excel

Библиотека ADO позволяет не только извлекать данные из книги Excel, но и вставлять и обновлять существующие записи так, как и при использовании других источников данных. Стоит отметить, что удаление записей не поддерживается. Обновление записей хотя и возможно, но связано с определенными сложностями, так как в таблицах данных на основе электронных таблиц Excel отсутствуют сущности, которые можно использовать в качестве первичного ключа для уникальной идентификации конкретных записей. Таким образом, в предложении WHERE необходимо указать значения достаточного количества полей, чтобы идентифицировать интересующую запись для обновления с помощью оператора SQL. Если критериям в предложении WHERE соответствует более одной записи, будут обновлены все соответствующие записи.

Вставка записей происходит намного проще. Достаточно создать запрос SQL, в котором указаны значения каждого поля, и отправить запрос поставщику. Обратите внимание, что для выполнения запросов на действие в таблице с данными должна присутствовать строка с названиями полей. В следующем примере показано, как вставлять новую запись в таблицу на листе:

```
Public Sub WorksheetInsert()

    Dim Connection As ADODB.Connection
    Dim ConnectionString As String

    ConnectionString = _
        "Provider=Microsoft.Jet.OLEDB.4.0;" & _
        "Data Source=" & ThisWorkbook.Path & "\Sales.xls;" & _
        "Extended Properties=Excel 8.0;"

    Dim SQL As String

    SQL = "INSERT INTO [Sales$] " & _
           "VALUES('VA', 'On Line', 'Computers', 'Mid', 30)"

    Set Connection = New ADODB.Connection
    Call Connection.Open(ConnectionString)

    Call Connection.Execute(SQL, , _
        CommandTypeEnum.adCmdText Or ExecuteOptionEnum.adExecuteNoRecords)

    Connection.Close
    Set Connection = Nothing
End Sub
```

Запросы для текстовых файлов

Последней в этой главе рассматривается метод доступа к данным в текстовых файлах средствами библиотеки ADO. Необходимость получать доступ к этим данным возникает намного реже, чем необходимость доступа к другим рассмотренным источникам данных, но при обработке исключительно большого текстового файла (например, дампа базы данных мейнфрейма) библиотека ADO может значительно упростить решение задачи.

Библиотека ADO не только поддерживает загрузку большого объема данных в Excel, но и позволяет использовать возможности SQL для ограничения объема набора записей, если текстовый файл оказывается слишком большим, чтобы непосредственно открывать его в Excel. Для обсуждения доступа к данным в текстовых файлах будет использоваться текстовый файл, в котором значения разделяются запятыми. Файл называется Sales.csv. Его

совпадает с содержимым файла Sales.xls, который использовался в предыдущих примерах работы с Excel.

Для доступа к данным в текстовых файлах также используется поставщик OLE DB для Microsoft Jet, но в этом случае применяется другая структура строки подключения. В следующем примере показано, как должна выглядеть строка подключения для доступа к данным в текстовом файле.

```
"Provider=Microsoft.Jet.OLEDB.4.0;" & _
             "Data Source=C:\Files\;" & _
             "Extended Properties=Text;"
```

Обратите внимание, что в случае работы с текстовыми файлами в качестве значения аргумента Data Source указывается имя каталога, в котором хранится интересующий текстовый файл. Имя файла в значении аргумента указывать не нужно. И в этот раз информация о формате передается поставщику через аргумент Extended Properties. В данном случае аргумент устанавливается в значение "Text".

Отправка запроса к текстовому файлу ничем не отличается от запроса к книге Excel. Основным отличием является формат именования таблицы в запросе SQL. В запросах к текстовым файлам в качестве имени таблицы используется имя файла, что позволяет работать с несколькими текстовыми файлами в одном каталоге, не меняя строки подключения.

Как и в случае с книгами Excel, если в первой строке текстового файла не хранятся названия полей, то при доступе к файлу поддерживаются только простые запросы SELECT. Кроме этого, в таком случае в аргумент Extended Properties необходимо добавить параметр HDR=No, иначе первая строка с данными окажется недоступна. В данном примере в первой строке файла хранятся названия полей. Предполагается, что для ограничения количества загружаемых в Excel данных, в запрос добавляется ограничение в виде предложения WHERE. Реализация такого поведения показана в следующей процедуре:

```
Public Sub QueryTextFile()
      Dim Recordset As ADODB.Recordset
      Dim ConnectionString As String
    ConnectionString = _
      "Provider=Microsoft.Jet.OLEDB.4.0;" & _
      "Data Source=" & ThisWorkbook.Path & ";" & _
      "Extended Properties=Text;"
      Const SQL As String = _
      "SELECT * FROM Sales.csv WHERE Type='Art';"
      Set Recordset = New ADODB.Recordset
    Call Recordset.Open(SQL, ConnectionString, _
      CursorTypeEnum.adOpenForwardOnly, _
      LockTypeEnum.adLockReadOnly, CommandTypeEnum.adCmdText)
      Call Sheet1.Range("A1").CopyFromRecordset(Recordset)
       Recordset.Close
    Set Recordset = Nothing
  End Sub
```

Резюме

На этом обсуждение доступа к данным средствами библиотеки ADO завершается. Ограниченный объем книги позволил только поверхностно ознакомиться с различными возможностями. Если доступ к данным является или может оказаться заметной частью разрабатываемого приложения, стоит обратиться к дополнительным источникам информации, упоминавшимся в этой главе.

12

Создание и использование надстроек

Надстройка (add-in) является фрагментом кода, который можно повторно использовать в Excel. Надстройка может быть реализована на языке Visual Basic 6 или в виде книги Excel, в которой хранится повторно используемый код. Например, имеет смысл повторно применить модули `DebugTools` и `EventLog` из книги `Bulletproof.xls`, которая рассматривалась в главе 7. В результате книгу `Bulletproof.xls` можно добавить в список надстроек и сделать ее недоступной для других книг. В этой главе будет показано, как превратить книгу в надстройку и сделать ее доступной для других книг.

В качестве надстроек могут выступать откомпилированные внешние приложения, хотя это и не обязательно. При этом, начиная с Office 2000 Developer Edition, поддерживается компиляция книг. Дополнительная информация об использовании откомпилированных надстроек приводится в главе 13.

В этой главе в качестве надстройки используется книга `Bulletproof.xls`. Здесь будет показано, как скрывать подробности реализации кода надстройки для предотвращения его случайного изменения.

Сокрытие кода

Книги являются интеллектуальной собственностью. Книги, которые используются в качестве надстроек, представляют интеллектуальную собственность, которую можно передавать или продавать другим разработчикам. При этом не обязательно предоставлять другим разработчикам информацию о подробностях реализации решения. Хотя невозможно скрыть факт существования надстройки, так как ее название отображается в окне Project Explorer (Окно проекта), однако надстройку можно защитить от просмотра, модификации и копирования кода.

Для сокрытия подробностей реализации интеллектуальной собственности исходный код можно защитить паролем. Для демонстрации книга `Bulletproof.xls` скопирована в папку с примерами для главы 12 и открыта в редакторе VBE. Для защиты исходного кода паролем (рис. 12.1) в редакторе VBE выберите Сервис⇨VBA Project (Tools⇨VBA Project), активизируйте вкладку Protection (Защита) и установите флажок Lock project for viewing (Защитить проект от просмотра). Введите пароль в соответствующее поле. Для доступа к исходному коду книги введите в качестве пароля слово **password**.

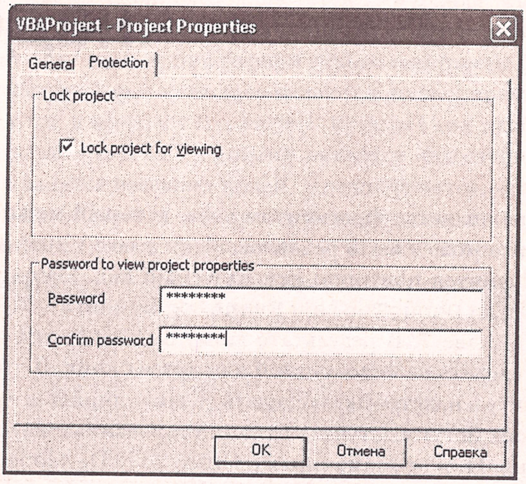


Рис. 12.1. Защита кода от просмотра

После ввода и подтверждения пароля щелкните на кнопке OK и сохраните файл. Для проверки эффективности пароля закройте файл и откройте его повторно. На рис. 12.2 показано, что ветка VBA Project в окне Project Explorer (Окно проекта) свернута и ее невозможно развернуть без ввода правильного пароля. Защита паролем является настолько надежной, насколько надежным является сам пароль и защищающий его механизм. Пароли в Excel не останавливают только самых настойчивых взломщиков.

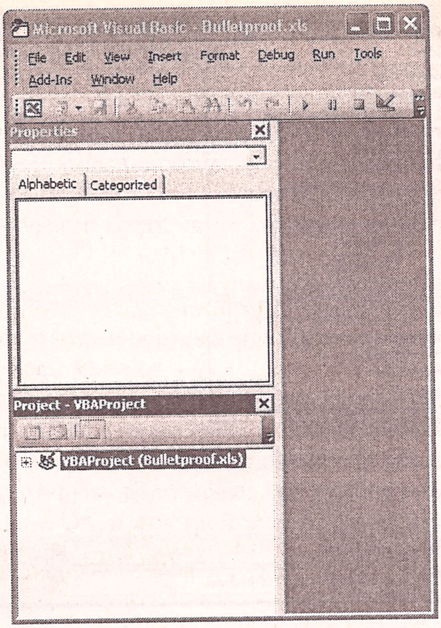


Рис. 12.2. Скрытый проект VBA

Преобразование книги в надстройку

> Для преобразования книги в надстройку необходимо переключиться в представление электронной таблицы и сохранить книгу с расширением `.xla`. Выберите меню Файл⇨Сохранить как (File⇨Save As) и в качестве типа файла выберите Microsoft Office Excel Add-in (*.xla) (рис. 12.3). При этом изменится расширение и надстройка будет сохранена в каталоге `C:\Document and settings\<имя_пользователя>\Application Data\Microsoft\Add-ins`. Изменение расширения с `.xls` на `.xla` не является обязательным, но это полезное соглашение, позволяющее потребителям различать файлы книг и файлы надстроек.

Альтернативным методом создания надстроек является установка свойства `IsAddin` объекта `ThisWorkbook` в значение `True`. Для этого можно воспользоваться окном Properties (Свойства) в редакторе.VBE (рис. 12.4). (Если перед этим книга была защищена паролем, для изменения свойств книги придется ввести пароль.)

Недостатком второго подхода является сохранение расширения файла `.xls`, хотя книга все равно превращается в надстройку. Для изменения расширения всегда можно воспользоваться Проводником Windows (Windows Explorer).

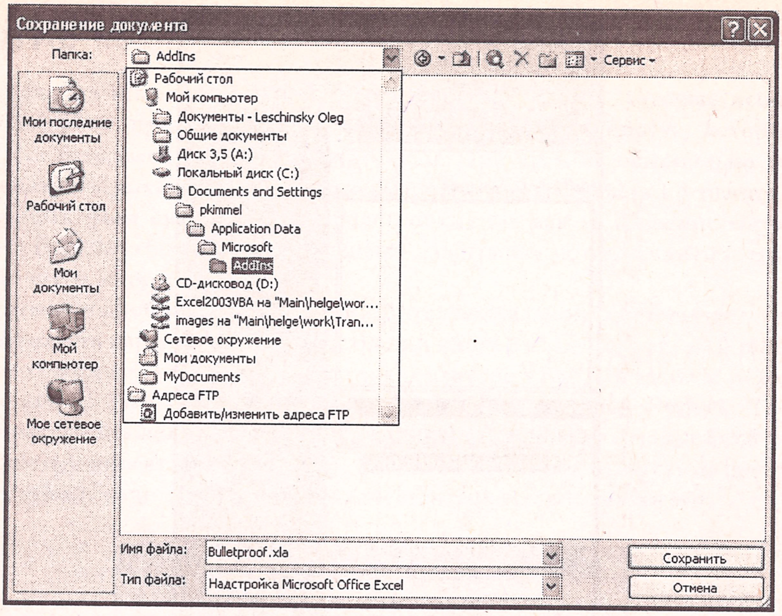


Рис. 12.3. Сохранение книги в качестве надстройки

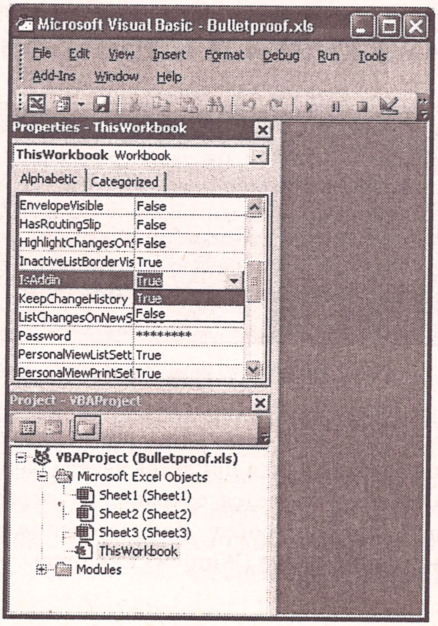


Рис. 12.4. Превращение книги в надстройку с помощью свойства IsAddin

Закрытие надстройки

Если книга была преобразована в надстройку с помощью установки свойства `IsAddin` с последующим сохранением или надстройка была загружена с помощью команды Файл⇨Открыть (File⇨Open), очевидного способа закрытия книги не существует, так как пункт Закрыть (Close) в меню Файл (File) отключен. Для обхода отключенного пункта в меню Файл (File) можно ввести следующую команду в окне Immediate (Проверка) в редакторе VBE.

```
Add-inWorkbooks("Add-inBulletproof.xls").Close
```

Надстройки не индексируются в коллекции `Workbooks` и не учитываются в значении свойства `Count` коллекции `Workbooks`.

Существует еще один метод закрыть надстройку. Для этого необходимо щелкнуть на имени недавно открытого файла в меню Файл (File), удерживая клавишу <Shift>. Может быть выдано сообщение о перезаписи копии в памяти (в зависимости от наличия изменений) и невозможности открыть надстройку для редактирования (атавизм последних версий). Щелкните на кнопке OK, и надстройка будет удалена из памяти.

Изменение кода

Иногда необходимо модифицировать код VBA, создававшийся для стандартной книги, чтобы подготовить его к использованию в надстройке. Это особенно справедливо, если приходится ссылаться на данные из надстройки. Большинство программистов пишут код, предполагающий, что книга и лист являются активными. Но ни один из компонентов надстройки не может быть активным, поэтому код должен явно ссылаться на книгу и лист надстройки. Предположим, что в надстройке содержится следующий код, рассчитанный для работы с активной книгой:

```
With Range("Database")
    Set Data = .Rows(2)
    Call LoadRecord
    Navigator.Value = 2
    Navigator.Max = .Rows.Count
End With
```

Этот код работает только в том случае, если диапазон `Database` находится в активной книге. В коде надстройки необходимо указать имя интересующих книги и листа. В данном случае для этого используется оператор `With`:

```
With Workbooks("workbook.xls").Sheets("Data").Range("Database").
```

Для ссылки на книгу, содержащую код, лучше использовать свойство `ThisWorkbook` объекта `Application`. Это свойство возвращает ссылку на книгу, содержащую выполняющийся код. Такая конструкция делает код значительно более гибким:

```
With ThisWorkbook.Sheets("Data").Range("Database")
```

Кроме этого, можно использовать имя объекта листа, которое отображается в окне Project Explorer (Окно проекта).

```
With Sheet1.Range("Database")
```

> В окне Properties (Свойства) можно редактировать программное имя листа и программное имя книги. Если изменить программное имя листа, то придется вносить изменения в код. Если изменить программное имя книги, можно использовать как новое имя, так и ссылку `ThisWorkbook`, которая остается действительной, так как является объектом `Application` и входит в коллекцию `<globals>`.

Если необходимо проигнорировать имя листа и позволить диапазону `Database` существовать на любом листе, можно воспользоваться следующей конструкцией:

```
With ThisWorkbook.Names("Database").RefersToRange
```

Сохранение изменений

Еще одной потенциальной проблемой, связанной с хранящими данные надстройками, является отсутствие автоматического сохранения изменений при завершении сеанса Excel. В частности, в данном примере допускаются изменения диапазона данных, который называется `Database`. Для сохранения изменений в надстройке добавьте следующий оператор в процедуру обработки события `BeforeClose` или `Auto_Close`.

```
If Not ThisWorkbook.Saved Then ThisWorkbook.Save
```

> Описанный прием не работает в Excel 5 и Excel 95. Эти версии не поддерживают сохранения изменений в файле надстройки.

Установка надстройки

Надстройку можно открыть из меню Файл (File), как было показано раньше. Но для получения большего контроля над надстройкой ее можно установить с помощью команды Сервис⇨Надстройки (Tools⇨Add-Ins). В результате выполнения этой команды открывается диалоговое окно, показанное на рис. 12.5.

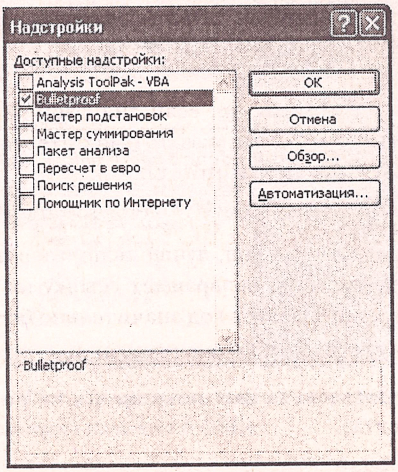


Рис. 12.5. Список надстроек, доступных для установки

Установите флажок напротив надстройки Bulletproof в списке доступных надстроек. Если надстройка отсутствует в списке, щелкните на кнопке Обзор (Browse) для поиска интересующей надстройки.

Дружественное название и описание предоставляются в свойствах книги. Если книга уже преобразована в надстройку, установите свойство `IsAddin` в значение `False`, чтобы сделать книгу видимой в Excel, и воспользуйтесь командой Файл⇨Свойства (File⇨Properties) для отображения следующего диалогового окна, показанного на рис. 12.6.

В поля Название (Title) и Комментарий (Comment) вносится информация, отображаемая в диалоговом окне Надстройки (Add-ins). После внесения необходимой информации можно установить свойство `IsAddin` в значение `True` и сохранить файл.

После того как надстройка появилась в диалоговом окне Сервис⇨Надстройки (Tools⇨Add-Ins), ее можно устанавливать и удалять, устанавливая и сбрасывая флажок возле названия надстройки. После установки надстройка загружается в память и становится видна в окне редактора VBE. Кроме этого, установленная надстройка загружается при последующих запусках Excel.

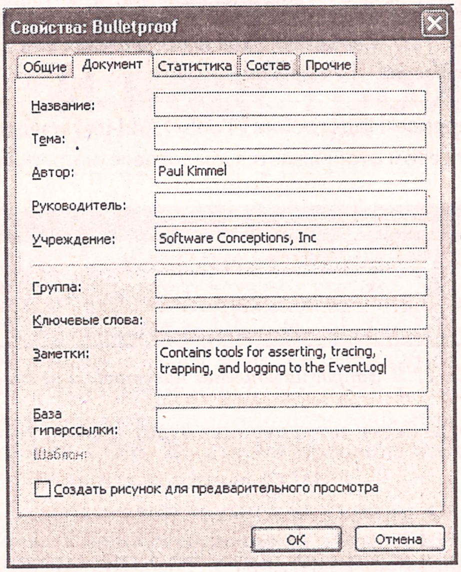


Рис. 12.6. Информация о надстройке

Событие установки надстройки

Существует два специальных события, которые возникают при установке и удалении надстройки. Следующий код, расположенный в модуле `ThisWorkbook`, выводит на экран диалоговое окно при каждой установке надстройки:

```
Private Sub Workbook_AddinInstall()
    InstallUserForm.Show
End Sub
```

Другим событием является `AddinUnistall`. Его можно использовать для отображения диалогового окна при удалении надстройки `Bulletproof`.

Удаление надстройки из списка надстроек

Одним из методов удаления надстройки является удаление файла из каталога `C:\Document and settings\<имя_пользователя>\Application Data\Microsoft\Add-ins` с помощью Проводника Windows (Windows Explorer) до открытия Excel. Альтернативным методом является переименование файла надстройки до открытия Excel. При запуске Excel после удаления или переименования выбранной ранее надстройки отображается показанное на рис. 12.7 диалоговое окно.

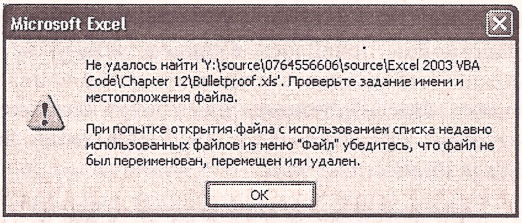


Рис. 12.7. Сообщение Excel после удаления или переименования надстройки

Выберите команду Сервис⇨Надстройки (Tools⇨Add-ins) и щелкните на флажке напротив названия надстройки. При этом отображается диалоговое окно, показанное на рис. 12.8.

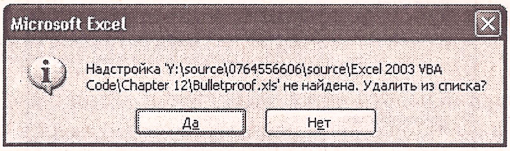


Рис. 12.8. Запрос на удаление надстройки из списка

Щелкните на кнопке Да (Yes), и надстройка будет удалена из списка установленных.

Резюме

Преобразование книги в надстройку позволяет распространять код среди других разработчиков, скрывая подробности реализации. В результате потребителям (включая самого разработчика) приходится использовать надстройку через открытые методы и свойства, не обращая внимание на подробности реализации. Это позволяет сконцентрироваться на проблеме и использовании существующего решения.

При создании надстройки необходимо модифицировать код, который ссылается на конкретные книги и предоставляет команды меню, элементы управления или кнопки панелей инструментов, предоставляющих доступ к макросам. Удаление ссылок на конкретные книги и листы было показано в этой главе. Связывание макросов с командами и кнопками панелей инструментов будет рассмотрено в главе 26.

13

Надстройки Automation и надстройки COM

Вместе с выходом пакета приложений Office 2000 компания Microsoft представила новую концепцию создания надстроек для всех приложений Office. Вместо создания надстроек для конкретных приложений (файлы `.xla` для Excel, файлы `.dot` для Word, файлы `.mde` для Access) с помощью Visual Basic, C++ или Office Developer Edition можно создавать библиотеки DLL, используемые в любом приложении Office. Так как эти библиотеки DLL соответствуют требованиям компонентной объектной модели Microsoft, они известны как надстройки COM. Во второй половине этой главы рассматривается создание и реализация собственных надстроек COM.

Надстройки COM обладают значительным недостатком. Содержащиеся в них функции невозможно использовать непосредственно на листе. В Excel 2002 и Excel 2003 компания Microsoft расширила концепцию и упростила механизм реализации надстроек COM, что позволило использовать подпрограммы надстроек в качестве функций на листе. Такие надстройки называются надстройками Automation.

Надстройки Automation

Надстройки Automation являются библиотеками COM DLL (ActiveX DLL). В библиотеках хранятся поддерживающие создание экземпляров классы и открытые функции. Как при работе с другими объектами, для вызова методов необходимо создать экземпляр класса. При этом используется немного более сложный синтаксис опосредованного вызова метода. Вместо именования объекта, метода и передачи параметра, информация передается через метод `CallByName`. Как минимум, необходимо создать объект COM и вызвать метод `CallByName`, передав в качестве аргументов объект COM, имя функции и параметры функции. В следующем примере показаны необходимые компоненты вызова:

```
Dim o As Object
Set o = CreateObject("идентификатор_объекта")
Call CallByName(o, "имя_функции", VbMethod, параметр1, параметр2,
            ... , параметр_n)
```

В этой главе будет рассмотрено несколько практических примеров, которые позволят получить необходимый опыт использования данного приема.

Для того чтобы объект COM стал объектом Automation, его необходимо создавать с определенными параметрами и он должен предоставлять как минимум один открытый метод. Для реализации некоторых примеров из этой главы необходима копия Visual Basic 6 (или VB.NET, Delphi, C++ или другой инструмент для создания изолированных исполнимых файлов Automation), но для использования объектов Automation дополнительные инструменты не нужны. Даже если изучение Visual Basic 6 или создание надстроек Automation не вызывает у вас интереса, все же просмотрите эту главу для понимания принципов создания расширений Excel.

Если нет возможности создавать объекты Automation, можно воспользоваться существующим приложением Automation, например Excel, Word, PowerPoint и SourceSafe.

Создание простой надстройки

Для разработчика VBA-приложений в Excel самым простым способом создания надстройки Automation является использование Visual Basic 6. К сожалению, такие надстройки невозможно создавать с помощью Office Developer Edition, так как в этом приложении не поддерживается создание классов `Public-Creatable`. (Параметр `Public-Creatable` описывает способ создания экземпляра класса. Объекты Automation создаются в соответствии со специальными правилами, а эти параметры заставляют компилятор добавлять необходимую информацию.) В этом примере с помощью Visual Basic 6 создается простая надстройка Automation. Для того чтобы основное внимание уделялось процессу, а не алгоритму, в данном случае определяется метод, возвращающий массив последовательных чисел. (Для компиляции этого примера потребуется Visual Basic 6, но существующий опыт в VBA позволит понять смысл кода.)

Для компиляции примера запустите Visual Basic 6 и создайте новый проект ActiveX DLL. Переименуйте проект в **MyAddin** и в окне Project (Проект) переименуйте класс в **Simple**. Установите свойство класса `Instancing` в значение `5-MultiUse`. Это значение принято по умолчанию для проектов ActiveX DLL. Установка этого свойства позволит Excel создавать экземпляры класса `Simple` и вызывать методы объекта.

В Visual Basic 6 класс определяется в модуле класса. В качестве примера модуля класса VBA можно привести модуль листа:

```
Option Base 1
Option Explicit

Public Function Sequence(ByVal Items As Long, _
  Optional ByVal Start As Integer = 1, _
  Optional ByVal Step As Integer = 1) As Variant()

' Невозможно создать массив с отрицательным количеством элементов
If Items < 1 Then
  Sequence = CVErr(2015)
  Exit Function
End If

Dim result As Variant
ReDim result(Items)
```

```
Dim I As Long

For I = 1 To Items
  result(I) = Start + I - 1 * Step
Next

  Sequence = result
End Function
```

Определение функции с квалификатором Public позволяет сделать ее доступной для Excel. В результате функцию можно будет вызывать из листа. Сохраните проект и воспользуйтесь командой редактора Visual Basic 6 Файл⇨Компилировать MyAddin.dll (File⇨Make MyAddin.dll) для создания библиотеки DLL. Результатом выполнения этой команды является надстройка Automation.

Регистрация надстроек Automation в Excel

Перед использованием функции Sequence на листе Excel необходимо сообщить Excel о существовании библиотеки DLL. Компания Microsoft расширила парадигму надстроек и включила в нее надстройки Automation. При этом применение надстроек Automation очень напоминает использование стандартных надстроек Excel .xla. Основным отличием является применение идентификатора класса ProgID вместо имени файла для идентификации надстроек Automation. Идентификатор класса состоит из имени проекта Visual Basic, точки и имени класса. В этом примере в качестве идентификатора ProgID класса Simple используется MyAddin.Simple.

Регистрация через пользовательский интерфейс Excel

Для использования надстройки в редакторе VBE необходимо создать ссылку на надстройку, экземпляр класса и вызвать интересующие методы надстройки. Откройте редактор VBE и выберите пункт Tools⇨References (Сервис⇨Ссылки). Найдите интересующую надстройку в списке Available References (Доступные ссылки). После обнаружения надстройки MyAddin установите флажок напротив названия надстройки. Щелкните на кнопке OK. Вот необходимая последовательность действий:

- загрузите Excel;
- нажмите комбинацию клавиш <Alt+F11> для переключения в редактор VBE;
- выберите пункт Tools⇨References (Сервис⇨Ссылки). Откроется диалоговое окно References (Ссылки);
- найдите надстройку MyAddin в списке Available References (Доступные ссылки);
- установите флажок и щелкните на кнопке OK.

После добавления ссылки на надстройку содержимое надстройки можно просматривать в окне Object Browser (Просмотр объектов). (Для доступа к окну Object Browser (Просмотр объектов) можно нажать клавишу <F2> или применить команду View⇨Object Browser (Вид⇨Просмотр объектов).) Теперь можно воспользоваться кодом из примера в начале этой главы и использовать возможности надстройки.

Создание ссылки на надстройку из кода VBA

Если на надстройку нужно сослаться из кода, ссылку можно создавать так же, как создаются ссылки на надстройки .xla. Для программного добавления надстройки Automation можно воспользоваться следующим кодом:

```
Sub InstallAutomationAddin()
    AddIns.Add Filename:="MyAddin.Simple"
    AddIns("MyAddin.Simple").Installed = True
End Sub
```

Добавление надстройки посредством редактирование системного реестра

Если код предназначен для распространения, изучите разделы и записи системного реестра, необходимые для ручной установки надстройки. Помните, что модификация системного реестра может вызвать нежелательные последствия, поэтому необходимо полное тестирование вносимых изменений (перед внесением изменений создайте копию системного реестра с помощью функции `Export`).

Надстройка модифицирует пару разделов системного реестра. Модифицируемые разделы и записи системного реестра рассматриваются ниже:

- запустите редактор системного реестра. Для этого выберите команду Пуск⇨Выполнить (Start⇨Run) и введите **regedit** или **regedt32**;
- перейдите в раздел системного реестра `HKEY_CURRENT_USER\Software\Microsoft\Office\11.0\Excel\Options`;
- создайте запись строкового типа Open. Для этого щелкните правой кнопкой мыши и выберите пункт Создать⇨Строковый параметр (New⇨String Value), после чего введите имя **Open** и значение **/A "MyAddin.Simple"**.

На рис. 13.1 показан примерный вариант содержимого реестра после внесения изменения. При следующем открытии Excel надстройка Automation `MyAddin` появится в списке доступных надстроек, который открывается по команде Сервис⇨Надстройки (Tools⇨Addins) (рис. 13.2).

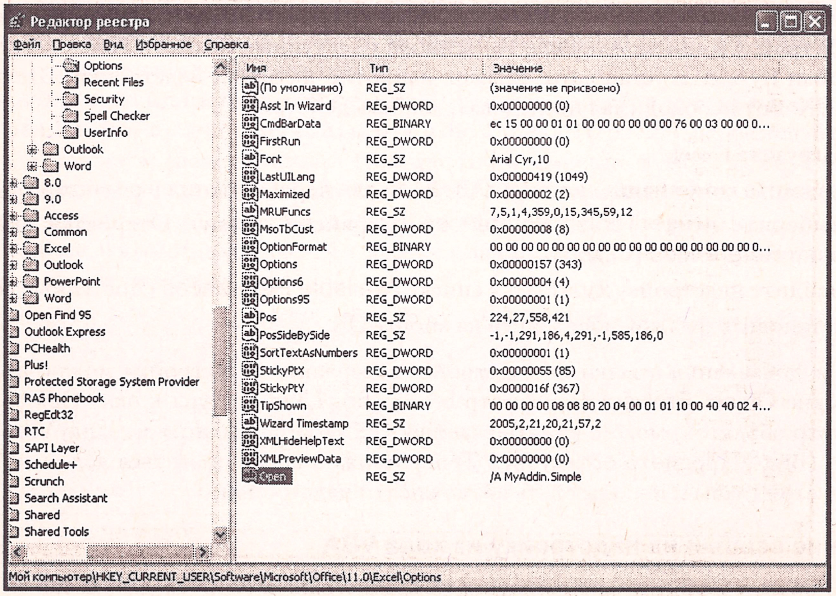


Рис. 13.1. Добавление параметра в системный реестр

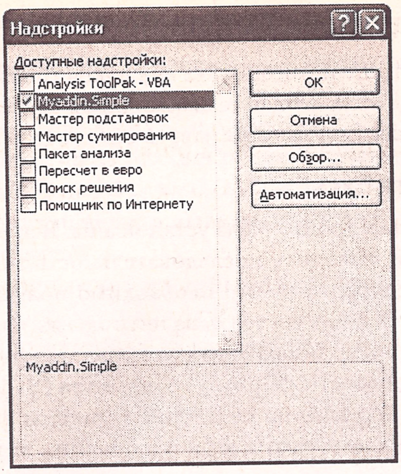


Рис. 13.2. Установка надстройки

Если надстройка должна быть доступна в диалоговом окне Надстройки (Addins), но не должна быть установлена, создайте раздел системного реестра `HKEY_CURRENT_USER\Software\Microsoft\Office\11.0\Excel\Addin Manager` и добавьте в этот раздел запись `MyAddin.Simple`. Это изменение позволит добавить в список доступных надстроек `MyAddin`, но при следующем запуске Excel надстройка не будет установлена (предполагается, что этот раздел не существовал раньше). После внесения изменений системный реестр будет выглядеть следующим образом (рис. 13.3).

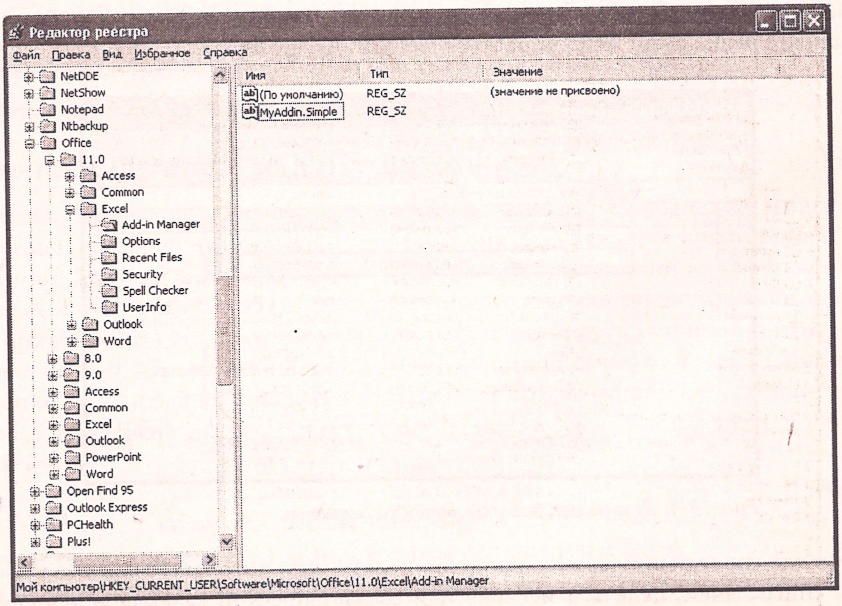


Рис. 13.3. Результат модификации системного реестра

Использование надстроек Automation

После добавления надстройки Automation код в надстройке можно воспринимать как класс. При этом поддерживается создание экземпляров класса и использование методов и свойств класса. Методы класса можно вызывать с листа Excel или из кода VBA.

Вызов функции из листа Excel

В предыдущем разделе надстройка была установлена. В этом разделе метод `Sequence` будет вызываться с листа и выводить последовательность значений — по одному значению в каждой ячейке. Предположим, что необходимо получить последовательность от 10 до 18. Последовательность будет состоять из пяти целых чисел начиная с 10 с шагом 2. Функцию `Sequence` можно использовать для заполнения пяти ячеек. Для этого необходимо выделить пять ячеек и ввести формулу **`=Sequence(5,10,2)`**. После ввода формулы нужно нажать комбинацию клавиш <Ctrl+Shift+Enter> и формула будет скопирована в каждую ячейку массива. После нажатия этой комбинации клавиш лист будет выглядеть, как показано на рис. 13.4.

Обратите внимание, что если имя функции в надстройке Automation вступает в конфликт со встроенной функцией Excel или функцией из надстройки `.xla`, Excel выбирает функцию в порядке приоритета. При этом наибольшим приоритетом обладает встроенная функция Excel, после которой следует функция из надстройки `.xla`. Функции из надстроек Automation обладают наименьшим приоритетом. Если возникает неожиданное поведение, вспомните о существовании приоритета.

Если Excel должна использовать функцию из конкретной надстройки, введите полностью квалифицированное имя функции, содержащее имя приложения Automation, имя класса и имя метода, например `=MyAddin.Simple.Sequence(5,10,2)`. После нажатия клавиши <Enter> Excel удалит имя надстройки Automation и имя класса, но эта информация все равно будет использоваться для выбора функции.

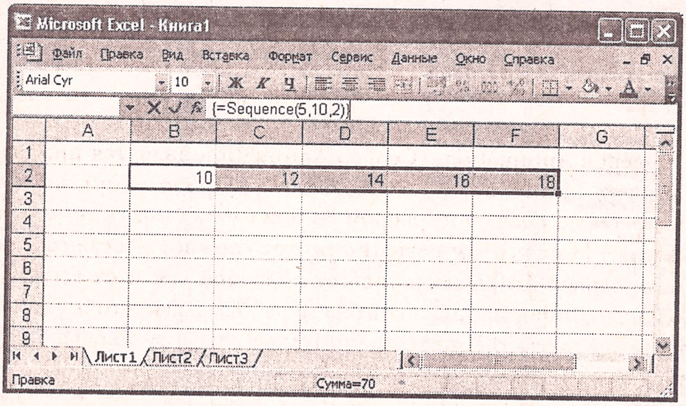


Рис. 13.4. Результат работы функции Sequence

В качестве упражнения можете написать надстройку, которая возвращает последовательный список дней, недель и месяцев или создает простой календарь для указанного месяца или года.

Вызов надстройки из кода VBA

Язык Visual Basic for Application предоставляет значительную гибкость. Воспользовавшись командой меню Tools⇨References (Сервис⇨Ссылки) можно ссылаться на надстройку в редакторе VBE и в коде VBA. Если функцию Sequence необходимо вызывать из процедуры обработки события Click для объекта CommandButton, добавьте кнопку на лист и в процедуре обработки события Click объявите и создайте экземпляр класса MyAddin.Simple. После этого можете вызвать процедуру Sequence. Такое решение показано в следующем листинге:

```
Private Sub CommandButton1_Click()
  ' Предполагается, что в меню
  ' Tools⇨References (Сервис⇨Ссылки) добавлена
  ' ссылка на надстройку Automation MyAddin
  Dim O As MyAddin.Simple
  Set O = New MyAddin.Simple
  ActiveCell.Resize(1, 5) = O.Sequence(5, 10, 2)

End Sub
```

Если точно известно, что надстройка установлена, используйте экземпляр, созданный Excel. Можно реализовать альтернативную версию метода CommandButton1_Click, в которой применяется метод Application.Evaluate:

```
Private Sub CommandButton1_Click()
    ActiveCell.Resize(1, 5) = _
        Application.Evaluate("MyAddin.Simple.Sequence(5,10,2)")
End Sub
```

При использовании метода Application.Evaluate полный идентификатор ProgID применяется только при конфликте имен со встроенными функциями или функциями из надстроек .xla. Если известно, что используются уникальные имена функций, можно не указывать имя надстройки Automation и имя класса, как показано ниже:

```
Application.Evaluate("MyAddin.Simple.Sequence(5,10,2)")
```

Введение в интерфейс IDTExtensibility2

До этого момента использовалась простая надстройка, так как она не зависит от других классов и серверов Automation. (Сервер Automation является приложением Automation. Термин "сервер" применяется для обозначения приложений, предоставляющих услуги. Excel можно также считать сервером Automation, так как ее услугами пользуются другие приложения.) Но с ростом решений растет сложность реализации. Это особенно справедливо, если сервер Automation требует обратной связи от Excel. Например, если необходимо получить информацию о контексте от вызывающего приложения, надстройка Automation должна знать о существовании Excel, а Excel должна знать о существовании надстройки.

В этом разделе рассматриваются свойства Application.Caller и Application.Volatile, а также реализация интерфейса IDTExtensibility2, обеспечивающего двухстороннюю связь между надстройкой Automation и Excel.

Для использования в надстройке Automation объекта Application из Excel ссылку на этот объект необходимо получить и сохранить в закрытой переменной внутри класса надстройки. Для этого в классе надстройки реализуется конкретный *интерфейс*.

С точки зрения синтаксиса интерфейс является объявлением без определений. Например, в объявлении интерфейса присутствует заголовок метода, но не указывается реализация. Интерфейс выступает в роли контракта, портала или грани. В реальном мире в качестве понятного всем примера можно привести интерфейс громкости. Если в интерфейсе `Громкость` объявлено два метода `Больше` и `Меньше`, любой класс, в котором реализован интерфейс `Громкость`, будет предоставлять определения методов `Больше` и `Меньше`, хотя каждая реализация будет отличаться от других. Например, телевизор, проигрыватель MP3 и стереоприемник (а также дети у счастливых родителей) могут содержать реализацию интерфейса `Громкость`.

В программном обеспечении реализация интерфейса определяет программный контракт. Если класс *реализует* конкретный интерфейс, он реализует и все компоненты этого интерфейса. Пример громкости можно расширить до реализации интерфейса `Громкость` в каждом устройстве и создания универсального пульта дистанционного управления, который запрашивает у устройств интерфейс `Громкость`. В этом случае можно управлять громкостью любого устройства.

В Excel встроена проверка конкретного интерфейса. При загрузке надстройки в Excel у нее запрашивается реализация интерфейса `IDTExtensibility2`. Если в надстройке реализован этот интерфейс, Excel может вызывать метод `OnConnection`, так как метод `OnConnection` является частью интерфейса `IDTExtensibility2`. Метод `OnConnection` позволяет Excel передавать надстройке ссылку на себя. Такая ссылка на объект Excel позволяет организовать двусторонний обмен данными. Excel может взаимодействовать с надстройкой через открытые методы, а надстройка может взаимодействовать с конкретным экземпляром Excel через полученную ссылку на объект Excel.

Для двунаправленного обмена данными ссылку на Excel, которая передается в надстройку через метод `OnConnection`, можно присвоить переменной. На самом деле интерфейс `IDTExtensibility2` предоставляет пять методов, каждый из которых должен быть реализован для удовлетворения контракта `IDTExtensibility2`. Хотя код для каждого из пяти методов необязательно предоставлять, как минимум придется создать пустой метод-заглушку.

В данном случае необходимо выполнить следующие задачи:

- создайте проект ActiveX DLL в Visual Basic 6;
- добавьте ссылку на библиотеку Microsoft Excel;
- реализуйте интерфейс `IDTExtensibility2`;
- объявите закрытую переменную `Excel.Application`, в которой будет храниться ссылка на объект Excel, передаваемая через метод `OnConnection`;
- реализуйте следующие четыре необходимых метода интерфейса `IDTExtensibility2`. Методы должны выполнять необходимые операции. Например, метод `OnDisconnection` может присваивать ссылке на Excel значение `Nothing`.

Рассмотрим каждый этап более подробно. Сначала необходимо создать библиотеку ActiveX DLL. Этот этап будет реализован в виде расширения предыдущего примера, `MyAddin`. После этого к существующему проекту нужно добавить новый класс (файл `.cls`). Назовите файл `Complex.cls`. В файле нового класса будет реализован интерфейс `IDTExtensibility2`.

После этого в Visual Basic 6 необходимо добавить ссылку на библиотеку Excel. Кроме этого, нужно добавить ссылку на библиотеку Microsoft Addin Designer. Вторая ссылка указывает на интерфейс IDTExtensibility2.

В качестве дополнительной операции в Visual Basic 6 необходимо добавить двоичную совместимость (binary compatibility). Это позволит новой библиотеке ActiveX DLL заменить предыдущую библиотеку DLL и записи в системном реестре вместо создания новых записей при каждом новом запуске. Для установки двоичной совместимости выберите команду Проект⇨Свойства MyAddin (Project⇨My Addin Properties), активизируйте вкладку Компоненты (Component) и выберите переключатель Двоичная совместимость (Binary Compatibility) (рис. 13.5).

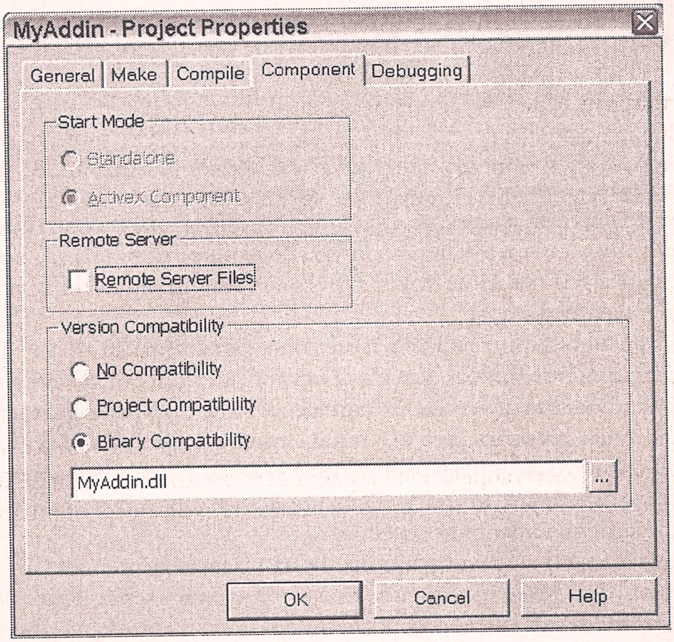


Рис. 13.5. Включение двоичной совместимости версий

После открытия диалогового окна Проект⇨Ссылки (Project⇨References), добавления библиотек Microsoft Addin Designer и Microsoft Excel 11.0 Object Library и включения двоичной совместимости можно начинать реализацию нового класса. Ниже приводится полный листинг и соответствующие пояснения к коду:

```
Implements IDTExtensibility2

Private Excel As Excel.Application

Private Sub IDTExtensibility2_OnConnection(ByVal _
    Application As Object, _
    ByVal ConnectMode As AddInDesignerObjects.ext_ConnectMode, _
    ByVal AddInInst As Object, custom() As Variant)

    Set Excel = Application
End Sub
```

```
Private Sub IDTExtensibility2_OnDisconnection( _
    ByVal RemoveMode As AddInDesignerObjects.ext_DisconnectMode, _
        custom() As Variant)

    Set Excel = Nothing
End Sub
Private Sub IDTExtensibility2_OnAddInsUpdate(custom() As Variant)
    ' Реализация намеренно опущена
End Sub

Private Sub IDTExtensibility2_OnBeginShutdown(custom() As Variant)
    ' Реализация намеренно опущена
End Sub

Private Sub IDTExtensibility2_OnStartupComplete(custom() As Variant)
    ' Реализация намеренно опущена
End Sub
```

В этом листинге предоставляется базовая реализация интерфейса IDTExtensibility2. В первом операторе листинга указывается реализуемый интерфейс (IDTExtensibility2). Редактор Visual Basic 6 помогает разработчику и генерирует пустые определения объявленных в интерфейсе методов. Во втором операторе объявляется переменная, в которой будет храниться ссылка на Excel. В первом методе OnConnection ссылка на экземпляр Excel копируется в эту переменную. Во втором методе OnDisconnection переменной присваивается значение Nothing, чтобы экземпляр Excel нельзя было использовать после завершения работы.

Оставшиеся три метода интерфейса намеренно оставлены пустыми. Контракт интерфейса требует создания методов-заглушек для каждого метода, объявленного в интерфейсе. Кроме этого, можно добавить все необходимые методы, включая реализации других интерфейсов. Достаточно помнить, что в данном случае создавший объект Complex экземпляр Excel будет видеть только интерфейс IDTExtensibility2. Все остальные интерфейсы будут невидимы для Excel, но могут использоваться для поддержки поведения интерфейса или предоставления нового поведения.

В данном случае стоит обратить внимание на существование надежного экземпляра приложения Excel. При этом надстройка может опрашивать Excel для более эффективного взаимодействия. В качестве эксперимента добавьте в метод OnConnection код, который будет запрашивать информацию об экземпляре Excel и возвращать некоторые данные. (Объектная модель Excel остается одинаковой при программировании “изнутри”, из Excel, и “снаружи”, из Visual Basic 6.) Далее еще будет рассмотрено более практическое применение интерфейса IDTExtensibility2.

Надстройка Complex — генерация уникального случайного числа

Теперь, когда получена ссылка на объект Excel Application, ее можно использовать для реализации более сложной функции. Показанная в следующем фрагменте кода функция Complex.GetNumbers возвращает случайный набор уникальных целых чисел из определенного диапазона. В этом методе объект Excel Application вызывается двумя разными способами.

- Функция Complex.GetNumbers использует свойство Application.Caller для определения диапазона ячеек, в котором расположен вызов функции. Эта информация позволяет определить размер и форму создаваемого массива.

- Функция Complex.GetNumbers использует вызов Application.Volatile для сброса генератора случайных чисел при каждом пересчете листа Excel.

Функция GetNumbers создает массив, соответствующий размерности выделенного диапазона. В каждый элемент массива вставляются уникальные случайные числа. После этого массив сортируется по значению элемента в порядке возрастания. Значения массива используются для заполнения выделенных ячеек листа.

> Представьте, что хотите выиграть в лотерею. Во многих региональных лотереях с высокими ставками необходимо угадать шесть чисел от 1 до 50. Здесь делается попытка определить минимальную ежедневную сумму денег, которая позволит покрыть максимальное количество вариантов. В результате был получен генератор случайных чисел, который генерирует восемь массивов по шесть уникальных чисел от 1 до 50. Это значит, что потратив $8, можно перекрыть 40 из 50 чисел. Теория вероятности подсказывает, что числа будут формировать случайные последовательности, и в итоге последовательности станут линейными последовательностями из 4, 5 и 6 (выигрышная комбинация!) совпадающих чисел. На самом деле этот прием использовался в течение шести недель, и каждую неделю генератор случайных чисел выдавал три выигрышных числа на одном билете и как минимум по одному числу на восьми билетах. (Интересно, какова вероятность того, что такая методика позволит повысить шансы на взятие джекпота? Теперь приходится жалеть, что в колледже теории вероятности и математической статистике уделялось так мало времени.)

Кроме этого, функция GetNumbers принимает необязательный параметр Items, позволяющий вызывать функцию как из ячейки листа, так и из кода VBA. Если предоставить параметр Items, функция возвращает двумерный массив (1,n) уникальных целых чисел. Если параметр Items не указывать, то для определения размерности массива функция воспользуется вызовом Application.Caller. Вот реализация генератора случайных чисел в Visual Basic 6:

```
Public Function GetNumbers(ByVal Min As Long, ByVal Max As Long)
↳ As Variant

    Dim aRange As Range
    Dim Values() As Double
    Dim Count As Long
    Dim I As Long
    Dim Value As Long
    Dim Rows As Long
    Dim Cols As Long

    Excel.Volatile

    Set aRange = Excel.Caller

    Rows = aRange.Rows.Count
    Cols = aRange.Columns.Count
    Count = Rows * Cols

    If (Not ValidateRange(Min, Max, Count)) Then
        GetNumbers = CVErr(xlErrValue)
        Exit Function '
    End If

    Values = GetRandomNumbers(Min, Max)
    Call Scramble(Values)

    GetNumbers = OrderValuesToCells(Rows, Cols, Values)
```

```
End Function

Private Function ValidateRange(ByVal Min As Long, _
    ByVal Max As Long, ByVal Count As Long) As Boolean
    ' Невозможно сгенерировать запрошенное количество случайных
    ' чисел, если количество возможных вариантов меньше
    ' количества запрошенных чисел
    Debug.Assert Min >= 0
    Debug.Assert Max > Min
    Debug.Assert Max - Min > Count

    ValidateRange = Max - Min > Count

End Function

Private Function GetRandomNumbers(ByVal Min As Long, _
    ByVal Max As Long) As Variant

    Dim Values() As Double
    Dim I As Long
    ReDim Values(1 To Max - Min, 1 To 2)

    ' Первое измерение массива заполняется возможными числами ,
    ' от min до max включительно.
    ' Второе измерение заполняется случайными числами.
    ' Эти числа будут использоваться
    ' для создания массива случайных чисел позднее.

    Randomize
    For I = 1 To Max - Min
        Values(I, 1) = I + Min - 1
        Values(I, 2) = Rnd
    Next I

    GetRandomNumbers = Values
End Function

Private Function OrderValuesToCells(ByVal Rows As Long, _
                ByVal Cols As Long, _
                ByVal Values As Variant) As Variant

    ' Отсортировать результаты таким образом, чтобы
    ' полученная конфигурация соответствовала конфигурации
    ' выделенного диапазона.
    Dim Results() As Double
    ReDim Results(1 To Rows, 1 To Cols)

    Dim R As Long
    Dim C As Long
    Dim Index As Long
    Index = LBound(Values)

    For R = 1 To Rows
        For C = 1 To Cols
            Results(R, C) = Values(Index, 1)
            Index = Index + 1
        Next C
    Next R

    OrderValuesToCells = Results
End Function
```

Изменение порядка случайных чисел

Метод GetNumbers возвращает двумерный массив чисел. В первой размерности содержатся все числа из диапазона от Min до Max. Во второй размерности содержатся случайные числа от 0 до 1. Первая размерность отсортирована, а вторая имеет случайный порядок чисел. (Для генерации случайных чисел можно использовать функцию Rnd, но при небольшом диапазоне могут возникать повторения. Данный метод генерации позволяет добиться отсутствия повторений.) Для изменения порядка уникальных чисел в диапазоне от Min до Max можно отсортировать случайные числа во второй размерности массива. Таким образом, для изменения порядка чисел в первой размерности необходимо отсортировать вторую размерность.

В подпрограмме Scramble реализован алгоритм пузырьковой сортировки, сравнивающий каждый элемент со всеми другими элементами и переносящий большие элементы в конец массива. Так как пузырьковая сортировка сравнивает каждый элемент со всеми другими элементами, в процессе сортировки выполняется n*n или n^2 сравнений. Это значит, что для относительно небольших значений n, например для n=1000, приходится выполнять большое количество сравнений. Тысяча элементов массива потребует 1000000 сравнений. По этой причине алгоритм пузырьковой сортировки используется достаточно редко. С другой стороны, современные персональные компьютеры обладают высокой производительностью и даже при использовании пузырьковой сортировки в состоянии очень быстро отсортировать от 10000 до 100000 элементов. В данном случае это достаточно быстро. (В качестве эксперимента выделите все ячейки на листе и введите GetNumbers(1,10000000). После этого нажмите <Ctrl+Shift+Enter>. Сортировка 10000000 элементов займет некоторое время.)

Если требуется быстрая сортировка большого набора данных, воспользуйтесь алгоритмами Selection Sort и QuickSort. Каждый из них хорошо подходит для сортировки определенного типа входных данных. Например, алгоритм QuickSort не намного быстрее SelectionSort или пузырьковой сортировки на массивах среднего размера. На относительно отсортированных массивах QuickSort может работать еще медленнее. Вот реализация алгоритмов Scramble/Sort и метода Swap:

```
Private Sub Scramble(ByRef Values As Variant)

    ' Здесь можно использовать простой алгоритм
    ' пузырьковой сортировки так как он хорошо
    ' справляется с массивами в 10000 элементов.
    ' Для сортировки массивов большего размера
    ' воспользуйтесь алгоритмами Selection Sort и QuickSort.

    Dim I As Long
    Dim J As Long

    For I = LBound(Values) To UBound(Values) - 1
        For J = I + 1 To UBound(Values)
            If (Values(I, 2) > Values(J, 2)) Then
                Call Swap(Values, I, J)
            End If
```

```
        Next J
    Next I

End Sub

Private Sub Swap(ByRef Values As Variant, ByVal I As Long,
↳ ByVal J As Long)

    Temp1 = Values(I, 1)
    Temp2 = Values(I, 2)

    Values(I, 1) = Values(J, 1)
    Values(I, 2) = Values(J, 2)
    Values(J, 1) = Temp1
    Values(J, 2) = Temp2

End Sub
```

После внесения всех изменений необходимо сохранить и перекомпилировать библиотеку MyAddin.dll в Visual Basic 6. Для этого воспользуйтесь командой Файл⇨Откомпилировать (File⇨Make). Если одновременно открыты Visual Basic 6 и Excel и в результате выбора этой команды выдается сообщение "В доступе отказано", закройте Excel, откомпилируйте надстройку и откройте ее в Excel повторно.

Надстройка Complex используется так же, как и показанная выше простая функция Sequence. Единственное различие заключается в том, что Excel необходимо сообщить о необходимости загрузить MyAddin.Complex. Для этого выберите команду Сервис⇨Надстройки⇨Надстройки Automation (Tools⇨Addins⇨Automation Addins) и надстройку из списка. При вводе формулы массива лист будет выглядеть следующим образом (рис. 13.6):

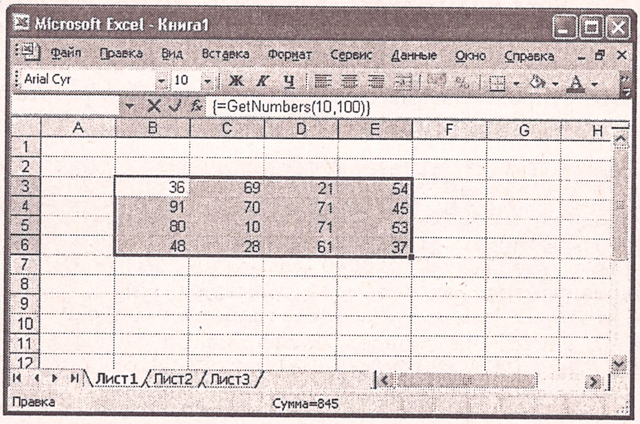


Рис. 13.6. Результат работы надстройки Complex

Как показано в этом примере, не стоит выделять весь лист для проверки работоспособности кода, так как при этом будет создан массив из более 16 миллионов случайных чисел. С таким объемом массива алгоритм пузырьковой сортировки просто не справится. В качестве альтернативы здесь приведена реализация сортировки с помощью алгоритма QuickSort. В данном случае для реализации алгоритма используется метод Swap:

```
Private Sub QuickSort(ByRef Values As Variant, _
    Optional ByVal Left As Long, Optional ByVal Right As Long)

    ' Алгоритм, использующий метод дихотомии, который хорошо
    ' справляется с большими не отсортированными массивами.

    Dim I As Long
    Dim J As Long
    Dim K As Long
    Dim Item1 As Variant
    Dim Item2 As Variant

    On Error GoTo Catch
    If IsMissing(Left) Or Left = 0 Then Left = LBound(Values)

    If IsMissing(Right) Or Right = 0 Then Right = UBound(Values)

    I = Left
    J = Right

    ' Получить элемент между Left и Right
    Item1 = Values((Left + Right) \bs  2, 2)

    ' Рассмотреть этот фрагмент массива значений
    Do While I < J

        Do While Values(I, 2) < Item1 And I < Right
            I = I + 1
        Loop

        Do While Values(J, 2) > Item1 And J > Left
            J = J - 1
        Loop

        If I < J Then
            Call Swap(Values, I, J)
        End If

        If I <= J Then
            I = I + 1
            J = J - 1
        End If
    Loop

    ' Рекурсивная обработка левого подмассива
    If J > Left Then Call QuickSort(Values, Left, J)

    ' Рекурсивная обработка правого подмассива
    If I < Right Then Call QuickSort(Values, I, Right)

    Exit Sub
Catch:
    MsgBox Err.Description, vbCritical

End Sub

Private Sub Swap(ByRef Values As Variant, ByVal I As Long, _
                ByVal J As Long)

    Dim Temp1 As Double
    Dim Temp2 As Double
```

```
    Temp1 = Values(I, 1)
    Temp2 = Values(I, 2)

    Values(I, 1) = Values(J, 1)
    Values(I, 2) = Values(J, 2)
    Values(J, 1) = Temp1
    Values(J, 2) = Temp2

End Sub
```

Если это интересно, выводите содержимое рассматриваемого массива на каждом этапе. При этом можно пронаблюдать, как массив рекурсивно разбивается на фрагменты и обмениваются пограничные значения. В результате получается отсортированный массив.

Надстройки COM

Хотя надстройки Automation позволяют создавать собственные функции листа, надстройки COM дают возможность расширять пользовательский интерфейс Excel и всех других приложений Office. Надстройки COM обладают рядом преимуществ по сравнению с обычными надстройками `.xla`:

- надстройки COM загружаются быстрее, чем надстройки `.xla`;
- надстройки COM не отображаются в окне Project Explorer (Окно проекта) в редакторе VBE;
- надстройки COM не могут быть модифицированы пользователями, так как являются откомпилированными двоичными файлами;
- надстройки COM не связаны непосредственно с Excel, а надстройки `.xla` могут использоваться только в Excel. Надстройки COM применимы в любом приложении Office.

Продолжение обзора интерфейса IDTExtensibility2

В предыдущем разделе было рассмотрено использование интерфейса `IDTExtensibility2`, в котором методы `OnConnection` и `OnDisconnection` применялись для получения ссылки на объект приложения Excel. Оставшиеся методы интерфейса могут использоваться надстройками COM для реагирования на конкретные события в процессе работы Excel. В следующей таблице рассматриваются все методы интерфейса `IDTExtensibility2`:

Метод	*Вызывается*	*Типичное применение*
`OnConnection`	При загрузке надстройки COM в Excel	Сохранение ссылки на приложение Excel, добавление пунктов меню в панели инструментов Excel и настройка процедур обработки событий
`OnStartupComplete`	После завершения загрузки всех надстроек и файлов в Excel	Отображение начального диалогового окна (как в Access и PowerPoint) или для изменения поведения в зависимости от присутствия других надстроек

Метод	*Вызывается*	*Типичное применение*
`OnAddInsUpdate`	При загрузке или выгрузке других надстроек COM	Если надстройка COM зависит от надстройки других надстроек, эта надстройка может выгрузить себя самостоятельно
`OnBeginShutdown`	В начале процесса завершения работы Excel	Остановка процесса завершения работы Excel в некоторых ситуациях или вызов процедур очистки перед завершением работы
`OnDisconnection`	При выгрузке надстройки COM в результате команды пользователя или в результате завершения работы Excel	Сохранение параметров. Если надстройка выгружается по команде пользователя, удалить элементы на панелях инструментов, созданные в процессе подключения

Как было показано ранее, реализация интерфейса требует предоставления реализации каждого метода интерфейса, даже если она является заглушкой.

Регистрация надстройки COM в Excel

Для уведомления Excel о существовании надстройки Automation необходимо выбрать команду Сервис⇨Надстройки⇨Надстройки Automation (Tools⇨Addins⇨Automation Addins). В результате использования открывшегося диалогового окна в системный реестр записывается соответствующая информация. Для регистрации в Excel надстройки COM в системном реестре необходимо создать соответствующие записи и разделы. При запуске Excel содержимое этих разделов используется для обнаружения доступных надстроек COM. Значения записей в разделах определяют отображение надстроек COM в списке надстроек и необходимость загрузки надстроек. Информация о надстройках COM для Excel хранится в следующих разделах системного реестра:

- надстройки, зарегистрированные для текущего пользователя: `HKEY_CURRENT_USER\Software\Microsoft\Office\Excel\Addins\AddinProgID`;
- надстройки, зарегистрированные для всех пользователей: `HKEY_USERS\.DEFAULT\Software\Microsoft\Office\Excel\Addins\AddinProgID`;
- надстройки, зарегистрированные для компьютера: `HKEY_LOCAL_MACHINE\Software\Microsoft\Office\Excel\Addins\AddinProgID`.

В каждый раздел добавляются следующие записи:

Имя	*Тип*	*Применение*
`FriendlyName`	`String`	Имя в списке надстроек COM
`Description`	`String`	Описание в диалоговом окне надстроек COM
`LoadBehavior`	`Number`	При выгрузке, загрузке при запуске или загрузке по требованию

Имя	*Тип*	*Применение*
`SatelliteDllName`	`Number`	Имя библиотеки ресурсов DLL, в которой содержатся локализованные имена и описания. При использовании такой библиотеки имя и описание будут указываться в виде `#Num`, где `Num` это номер идентификатора ресурса в библиотеке ресурсов. Этот прием используется для локализации большинства надстроек Office
`CommandLineSafe`	`String`	Определяет возможность вызова надстройки из приглашения интерпретатора командной строки. Эта запись не имеет значения для библиотек COM

После правильной регистрации надстройка COM станет доступна в диалоговом окне Excel Надстройки COM (COM Addins). В этом окне можно загружать и выгружать надстройки COM как обычные. К сожалению, команда для отображения этого диалогового окна отсутствует в стандартных меню Excel. Для получения доступа к этому диалоговому окну необходимо модифицировать панели инструментов.

- Щелкните правой кнопкой мыши на панели инструментов Excel и выберите пункт Настройка (Customize).
- Щелкните на пункте меню Сервис (Tools) для просмотра списка подменю.
- В диалоговом окне Настройка (Customize) активизируйте вкладку Команды (Commands), выберите пункт Сервис (Tools) в списке слева и прокрутите правый список, пока не обнаружите пункт Надстройки COM (COM Addins) (рис. 13.7).
- Перетащите пункт Надстройки COM (COM Addins) из правого списка в панель инструментов Сервис (Tools) под пунктом меню Надстройки (Addins).
- Закройте диалоговое окно Настройка (Customize).

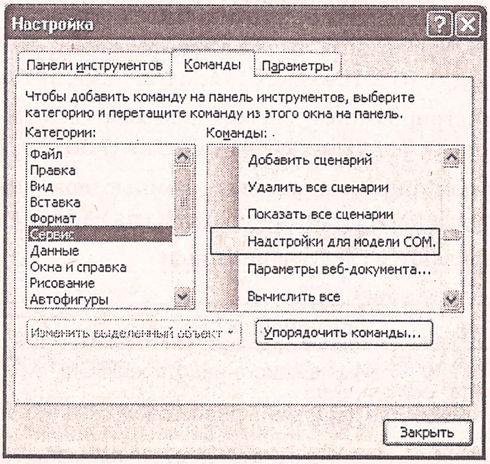


Рис. 13.7. Добавление пункта меню в меню Сервис (Tools)

Конструктор надстроек COM

Компания Microsoft предоставляет класс `Designer`, который можно использовать для упрощения создания и регистрации надстроек COM. В классе `Designer` реализован интерфейс `IDTExtensibility2`. Методы класса `Designer` вызываются через события, поэтому предоставление методов-заглушек не требуется. Для использования класса `Designer` достаточно реализовать процедуры обработки интересующих событий. Класс `Designer` предоставляет графический интерфейс, упрощающий ввод значений в записи системного реестра. При компиляции класса `Designer` выполняется автоматическое добавление кода (точки входа `DllRegisterServer`), который добавляет все необходимые разделы системного реестра. (По умолчанию разделы записываются только для текущего пользователя `HKEY_CURRENT_USER`.) Для регистрации сервера достаточно запустить команду `regsvr32 application.dll`, где вместо `application.dll` необходимо указать путь и имя файла библиотеки COM. Пакет Office Developer Edition позволяет создавать и компилировать надстройки COM в VBE, а не в Visual Basic 6. (Для создания надстройки в редакторе VBE из Developer Edition выберите команду Файл⇨Создать⇨Проект надстройки (New⇨Project⇨Addin Project).)

В этом примере создается надстройка COM, предоставляющая мастера для ввода функции `GetNumbers` из надстройки Automation, созданной в предыдущем разделе. Для создания библиотеки `MyAddin.dll` в данном случае также будет использоваться Visual Basic.

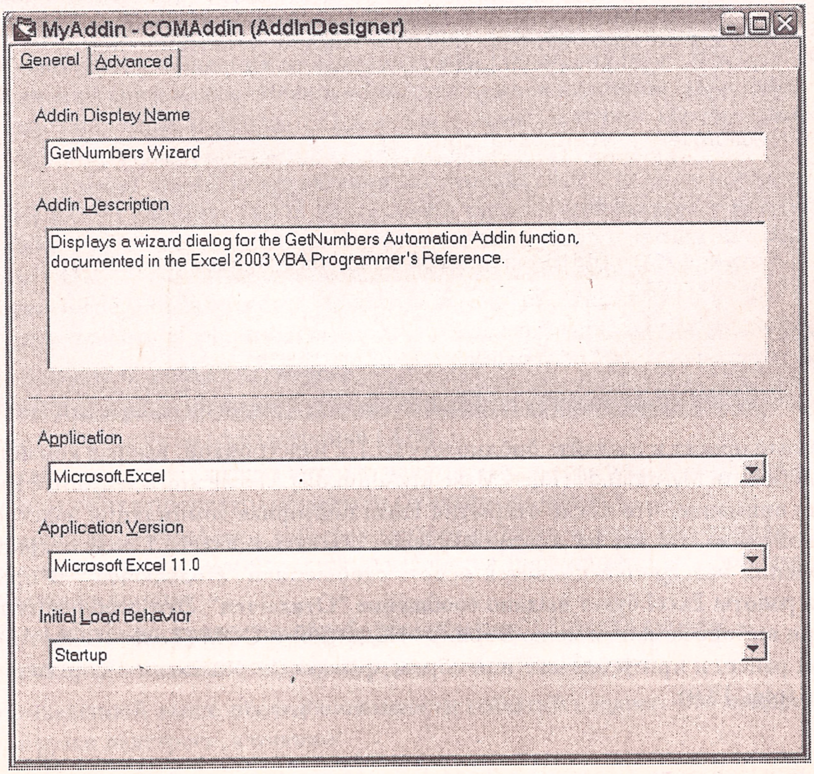


Рис. 13.8. Свойства надстройки

Откройте проект MyAddin в Visual Basic. Добавьте в проект новый класс надстройки. Для этого выберите Проект⇨Добавить класс надстройки (Project⇨Add Addin Class). (Если эта команда недоступна, выберите команду Проект⇨Компоненты⇨Конструкторы (Project⇨Components⇨Designers) и установите флажок напротив пункта Класс надстройки (Addin Class)). После этого будет добавлен класс Designer, который называется AddinDesigner1. В окне Свойства (Properties) измените имя класса на COMAddIn и установите свойство Public в значение True (не обращайте внимание на предупреждения). Заполните поля в окне Конструктор (Designer), как показано на рис. 13.8.

Подключение к Excel

Выберите команду COMAddin в окне Project Explorer (Окно проекта) в Visual Basic 6. Щелкните правой кнопкой мыши на конструкторе COMAddIn, созданном в предыдущем разделе. После этого реализуйте процедуры обработки событий OnConnection и OnDisconnection, выбрав AddinInstance из раскрывающегося списка Object (Объект) и процедуры OnConnection и OnDisconnection из раскрывающегося списка Procedure (Процедура). Добавьте оператор WithEvents, в котором объявляется переменная типа Excel.Application. Другими словами, здесь сохраняется ссылка на вызывающее приложение Excel, которая передается через процедуру обработки события OnConnection:

```
Private WithEvents Excel As Excel.Application

Private Sub AddinInstance_OnConnection(ByVal Application As Object, _
  ByVal ConnectMode As AddInDesignerObjects.ext_ConnectMode, _
  ByVal AddInInst As Object, custom() As Variant)

  Set Excel = Application
  MsgBox "Подключен", vbInformation

End Sub

Private Sub AddinInstance_OnDisconnection( _
  ByVal RemoveMode As AddInDesignerObjects.ext_DisconnectMode, _
  custom() As Variant)

  Set Excel = Nothing
  MsgBox "Отключен", vbInformation

End Sub
```

Сохраните проект и создайте библиотеку DLL с надстройкой, выбрав команду Файл⇨Откомпилировать MyAddin.dll (File⇨Make MyAddin.dll). После этого запустите Excel 2003. (Обратите внимание, что после открытия Excel повторная компиляция надстройки окажется невозможна, так как на надстройку будет ссылаться Excel.) После открытия Excel и подключения надстройки появится окно с сообщением "Подключен". В процессе завершения работы Excel будет выдано сообщение "Отключен". Эти сообщения также выдаются при выборе команды меню Сервис⇨Надстройки COM (Tools⇨COM Addins), добавленной ранее, и при установке или сбросе флажка GetNumbers в списке доступных надстроек (рис. 13.9).

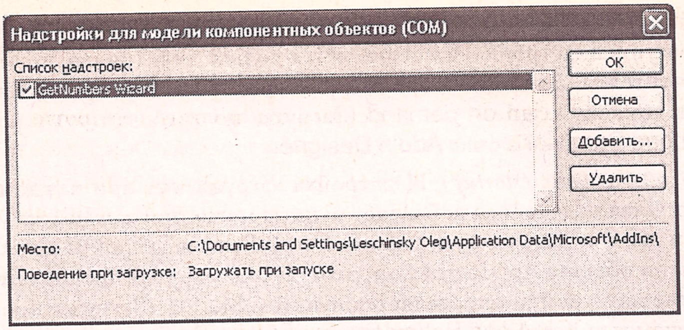


Рис. 13.9. Установка надстройки COM

Обработка событий Excel

Модуль кода Designer является модулем класса. Это значит, что с помощью оператора WithEvents можно определить переменную, позволяющую создавать процедуры обработки событий класса. В показанном ранее коде выполнялось подключение к событиям объекта Excel Application, что позволяло надстройке COM реагировать на открытие и закрытие надстроек пользователями, изменение данных в ячейках и на другие события Excel. Дополнительная информация об этих событиях была приведена в главе 12.

Добавление элементов управления и панелей инструментов

После получения ссылки на объект Excel Application собственные панели инструментов и элементы управления можно добавлять так же, как будет показано в главе 26. Единственная разница заключается в обработке щелчка на кнопке. При добавлении элемента управления CommandBarButton из Excel свойству OnAction присваивается имя процедуры, которая запускается при щелчке на кнопке.

При добавлении элемента управления CommandBarButton из-за пределов Excel (из надстройки COM) свойству OnAction объекта кнопки присваивается указатель на надстройку COM (в результате Excel будет знать, что за обработку щелчка на кнопке отвечает надстройка COM). После этого привязка к событию Click объекта кнопки осуществляется через переменную, объявленную с помощью оператора WithEvents в коде надстройки. Вот последовательность событий и вызовов, возникающих при щелчке на кнопке:

- пользователь щелкает на кнопке;
- Excel проверяет значение свойства OnAction объекта кнопки и считывает значение свойства ProgId надстройки COM;
- Excel проверяет, загрузилась ли надстройка. Если нет, надстройка загружается и запускается процедура обработки события OnConnection;
- в процедуре обработки события OnConnection в коде надстройки с помощью ключевого слова WithEvents объявляется переменная, которой присваивается ссылка на кнопку;
- Excel запускает процедуру обработки события Click для кнопки;
- в надстройке выполняется процедура обработки события.

Из приведенной выше последовательности событий следует, что существует два варианта загрузки надстроек, а именно:

- *загрузка по требованию (demand-loaded)*. Excel загружает надстройку при первой регистрации. В панели меню добавляются пункты меню надстройки с установкой свойства OnAction. При закрытии Excel эти пункты меню сохраняются. При сле-

дующей загрузке Excel надстройка загружается только при щелчке на пункте меню. Если доступ к надстройке осуществляется только через меню, то этот метод более предпочтителен. Для включения загрузки надстройки по требованию необходимо выбрать команду Load on demand (Загрузка по требованию) в раскрывающемся списке Load Behavior в окне Addin Designer;

- *загрузка при запуске (startup).* Надстройка загружается при каждом запуске Excel. Обычно такая надстройка добавляет пункты меню при каждом запуске Excel и удаляет при каждом завершении работы Excel. Если надстройка должна реагировать на события объекта Application, этот метод загрузки предпочтительней. Такое поведение надстройки определяется пунктом Startup (Загрузка при запуске) в раскрывающемся списке Load Behavior в окне Addin Designer.

В следующем примере будут добавлены два пункта меню, отображающие диалоговые окна мастера для поддержки ввода формул надстроек Automation. Для использования объектов CommandBarButton потребуется ссылка на объектную библиотеку Office. Для этого щелкните на пункте Project⇨References (Проект⇨Ссылки) и установите флажок напротив Microsoft Office 11.0 Object Library.

Модифицируйте объявления в коде надстройки COM. Добавьте объявление переменных CommandBarButton и String. При этом будут объявлены переменные уровня класса, которые будут использоваться для хранения ссылки на объект Excel Application и подключения к событиям объекта CommandBarButton:

```
Private WithEvents Excel As Excel.Application
Private WithEvents MenuButton As Office.CommandBarButton
Const AddInTag As String = "MyAddinTag"
```

При подключении к событиям объекта CommandBarButton с помощью ключевого слова WithEvents переменная MenuButton связывается со свойством Tag объекта кнопки, на которую ссылается переменная. Все кнопки с одинаковым значением свойства Tag будут запускать процедуру обработки события Click. При этом событие Click для всех кнопок с одинаковым значением свойства Tag можно перехватить с помощью единственной переменной WithEvents. Для того чтобы различать кнопки, в методе OnConnection каждой кнопке присваивается уникальное значение свойства Parameter. Ниже показаны модифицированные методы OnConnection и OnDisconnection:

```
Private Sub AddinInstance_OnConnection(ByVal Application As Object, _
    ByVal ConnectMode As AddInDesignerObjects.ext_ConnectMode, _
    ByVal AddInInst As Object, custom() As Variant)
        Set Excel = Application
    MsgBox "Connected", vbInformation

    Call InsertToolButton(AddInInst)
End Sub

Private Sub InsertToolButton(Byval AddInInst As Object)
    Dim Toolbar As CommandBar
    Dim Button As CommandBarButton

    Set Toolbar = Excel.CommandBars("Worksheet Menu Bar") _
        .FindControl(ID:=30007).CommandBar

    On Error Resume Next

    Set Button = Toolbar.Controls("Sequence Wizard")
```

```
    If (Button Is Nothing) Then
        Set Button = Toolbar.Controls.Add(msoControlButton, ,
↳ "SequenceWizard")
        Button.Caption = "Sequence Wizard"
        Button.Style = msoButtonCaption
        Button.Tag = AddInTag
        Button.OnAction = "!<" & AddInInst.ProgId & ">"
    End If

    Set Button = Nothing
    If (Button Is Nothing) Then
        Set Button = ToolBar.Controls.Add(msoControlButton, ,
↳ "GetNumbersWizard")
        Button.Caption = "GetNumbers Wizard"
        Button.Style = msoButtonCaption
        Button.Tag = msAddinTag
        Button.OnAction = "!<" & AddInInst.ProgId & ">"
End If

    Set MenuButton = Button

End Sub
```

Обратите внимание, что значение свойства `OnAction` должно иметь специальный формат, иначе Excel не воспримет значение в качестве ссылки на надстройку COM. Ссылка должна выглядеть как `"!<ProgID>"`.

Ниже показан пример типичной надстройки с загрузкой по требованию. Пункты меню данной надстройки удаляются, только если пользователь выгружает надстройку в диалоговом окне **Надстройки COM** (COM Addin). Метод выгрузки определяется по значению свойства `RemoveMode`. Пункты меню загружаемых при запуске надстроек удаляются при выгрузке надстройки:

```
Private Sub AddinInstance_OnDisconnection( _
    ByVal RemoveMode As AddInDesignerObjects.ext_DisconnectMode, _
    custom() As Variant)

    Dim Control As CommandBarControl

    ' Удалить кнопки, если Excel закрывается пользователем
    If RemoveMode = ext_dm_UserClosed Then
        For Each Control In Excel.CommandBars.FindControls(Tag:=AddInTag)
        Control.Delete
        Next
    End If

    Set MenuButton = Nothing
    Set Excel = Nothing
    MsgBox "Отключен", vbInformation

End Sub
```

В процедуре обработки события `Click` объекта `MenuButton` проверяется значение свойства `Parameter` той кнопки, на которой щелкнул пользователь. После этого выводится диалоговое окно, связанное с данной кнопкой. На этом этапе к проекту достаточно добавить два пустых диалоговых окна, назвать их `SequenceWizardForm` и `GetNumbersWizardForm`:

```
Private Sub MenuButton_Click( _
    ByVal Ctrl As Office.CommandBarButton, CancelDefault As Boolean)
```

```
    If TypeOf Excel.Selection Is Range Then
        Select Case Ctrl.Parameter
            Case "SequenceWizard"
            SequenceWizardForm.Show vbModal

            Case "GetNumbersWizard"
                GetNumbersWizardForm.Show vbModal
            End Select
    Else
        MsgBox "Не выбран диапазон ячеек.", vbOKOnly, _
                "Мастер Excel 2003"
    End If
End Sub
```

Сохраните проект и воспользуйтесь командой Файл⇨Компилировать MyAddin.dll (File⇨Make MyAddin.dll) для создания библиотеки DLL. При этом для Excel будут добавлены записи системного реестра. Запустите Excel 2003 и выберите команду Сервис⇨Sequence Wizard (Tools⇨Sequence Wizard). После этого будет показано диалоговое окно мастера.

Использование надстройки COM из кода VBA

Разработчик надстройки COM может предоставить программный доступ к надстройке из кода VBA (к сожалению, так бывает не часто). Это может потребоваться для:

- предоставления функциональности надстройки коду сторонних разработчиков;
- предоставления механизма для управления и модификации надстройки.

Для предоставления механизма управления надстройкой из кода VBA свойству `Object` экземпляра надстройки необходимо присвоить ссылку на класс надстройки COM (или отдельный класс в пределах надстройки). После этого необходимая функциональность предоставляется через свойства и методы, объявленные с квалификатором `Public`. В данном примере предоставляется еще один способ использования функций `Sequence` и `GetNumbers`.

Добавьте следующую строку кода в конец подпрограммы `AddinInstance_OnConection` для предоставления ссылки на класс надстройки через свойство `Object`:

```
AddInInst.Object = Me
```

После этого добавьте следующий код в конец модуля класса `Designer`. Этот код создает и возвращает новые экземпляры классов `Simple` и `Complex`:

```
Public Property Get SimpleObject() As Simple
  Set SimpleObject = New Simple
End Property

Public Property Get ComplexObject() As Complex
  Set ComplexObject = New Complex
End Property
```

Данный код в Excel позволяет получать доступ к функции `Sequence`. Для этого используется надстройка COM и свойство `Object`.

```
Private Sub CommandButton1_Click()

    Dim Sequence As Variant
    Sequence = Application.COMAddIns( _
        "MyAddin.COMAddIn").Object.SimpleObject.Sequence(5, 10, 2)
    ActiveCell.Resize(1, 5) = Sequence

End Sub
```

Главной особенностью этого подхода является применение того же экземпляра класса надстройки, что и в Excel. Это позволяет манипулировать, опрашивать и управлять надстройкой из кода VBA. В случае более сложных надстроек COM этот же метод может использоваться для предоставления доступа к полной объектной модели для управления надстройкой.

Связывание с несколькими приложениями Office

В начале этой главы отмечалось, что одним из главных преимуществ надстроек COM по сравнению с надстройками `.xla` является возможность использования одной библиотеки DLL в нескольких приложениях Office. Для этого достаточно добавить класс надстройки `Designer` в каждое интересующее приложение. Добавление класса `Designer` для Excel было показано ранее в этой главе. Конечно, особенности каждого приложения придется обрабатывать отдельно.

В следующем простом примере функция `Sequence` будет предоставлена через надстройки COM в Access и использована для заполнения списка в диалоговом окне.

Начните с добавления в проект нового класса надстройки. В окне Свойства (Properties) измените имя класса на `AccessAddin`, установите свойство `Public` в значение `True` (не обращайте внимание на предупреждения). Внесите информацию в диалоговое окно Designer, как показано на рис. 13.10.

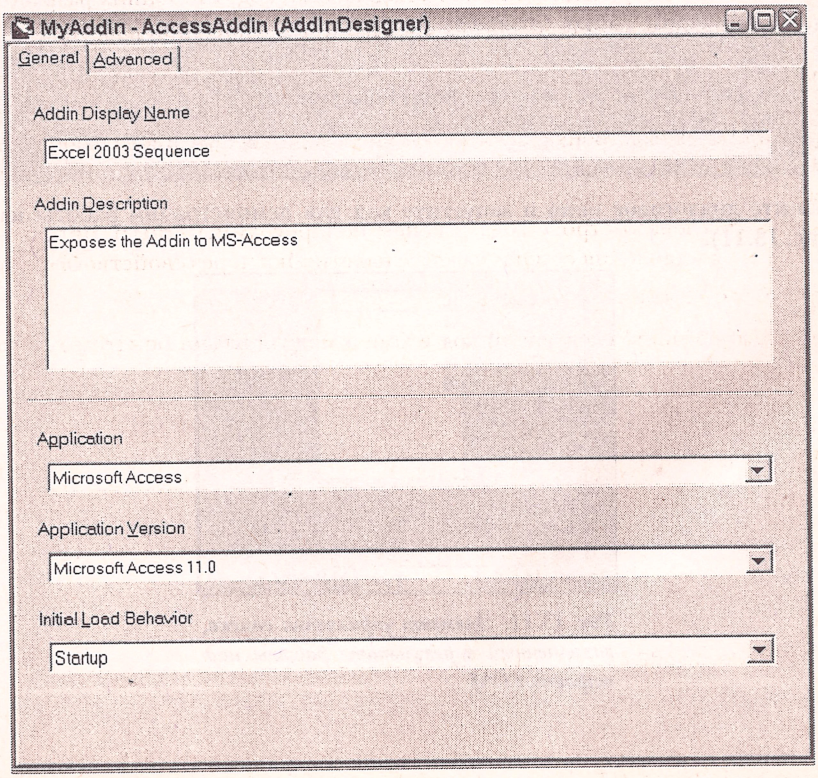


Рис. 13.10. Настройка свойств надстройки

Выберите команду Вид⇨Код (View⇨Code) и скопируйте следующий код в модуль кода Designer:

```
Private Sub AddinInstance_OnConnection(ByVal Application As Object, _
    ByVal ConnectMode As AddInDesignerObjects.ext_ConnectMode, _
    ByVal AddInInst As Object, custom() As Variant)
        AddInInst.object = Me

End Sub

Public Property Get SimpleObject() As Simple
    Set SimpleObject = New Simple
End Property
```

Сохраните проект и воспользуйтесь командой Файл⇨Откомпилировать MyAddin.dll (File⇨MyAddin.dll) для компиляции библиотеки DLL. Запустите Access 2003 и создайте пустую базу данных. Создайте новое диалоговое окно, добавьте в него элемент управления списка и скопируйте следующий код в модуль кода диалогового окна:

```
Option Explicit
Option Compare Database

Private Sub Form_Load()
    Dim Sequence As Variant
    Dim I As Integer

    Sequence = Application.COMAddIns( _
        "MyAddin.AccessAddin").Object.SimpleObject.Sequence(5, 10, 2)

    For I = LBound(Sequence) To UBound(Sequence)
        List0.AddItem Sequence(I)
    Next
End Sub
```

Сохраните диалоговое окно и запустите код для демонстрации работы надстройки COM (рис. 13.11).

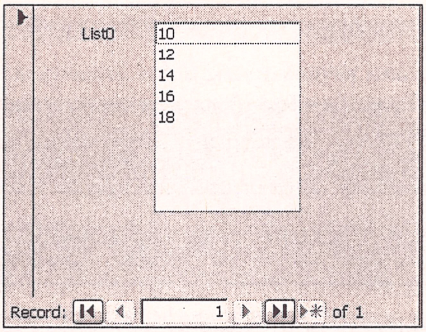


Рис. 13.11. Элемент управления списка, заполненный в результате работы надстройки COM

Резюме

Вместе с Excel 2003 компания Microsoft предоставила несколько способов использования надстроек, написанных на Visual Basic или любом другом языке, поддерживающем создание библиотек COM DLL.

- Надстройки Automation позволяют добавлять новые функции, которые будут доступны на листах и в процедурах VBA.
- Надстройки COM позволяют добавлять новые пункты меню и поддерживают обработку событий Excel. Кроме этого, такие надстройки можно использовать в нескольких приложениях Office и в редакторе VBE.
- Надстройки COM обеспечивают программный доступ к поведению надстройки, например поддерживают включение и отключение операций или использование функций надстройки.
- Надстройки Automation и COM обычно обеспечивают большую производительность, чем надстройки VBA.

В главе 27 будет рассмотрен третий способ расширения функциональности Excel через библиотеки ActiveX DLL — смарт-теги.

14

Настройка редактора VBE

В списке библиотек в диалоговом окне Tools⇨References (Сервис⇨Ссылки) редактора VBE присутствует объектная библиотека Microsoft Visual Basic for Applications Extensibility 5.3. Свойства, методы и события объектов из этой библиотеки позволяют:

- программно создавать, удалять и модифицировать код, диалоговые окна `UserForm` и ссылки на собственные и чужие книги;
- программировать редактор VBE для создания надстроек, помогающих при разработке и автоматизирующих многие задачи программирования.

С выходом Office 2000 объекту `CommandBarButton` было добавлено событие `Click`, используемое для обработки щелчков на кнопках, добавленных в панели инструментов редактора VBE. Таким образом, подобная надстройка не будет работать в Excel 97, хотя код управления редактором VBE и его объектами остается актуальным.

Единственное отличие Excel 2002 от Excel 2000 связано с механизмом обеспечения безопасности. Показанные в этой главе методы используются макровирусами для модификации кода целевого файла, что приводит к инфицированию, для предотвращения которого компания Microsoft позволила отключать доступ к проектам книг. Доступ запрещен по умолчанию, поэтому код из этой главы работать не будет. Для предоставления доступа к проектам установите флажок Доверенный доступ к проекту Visual Basic (Trust Access to Visual Basic Project) в окне Сервис⇨Макрос⇨Безопасность⇨Доверенные источники (Tools⇨Macros⇨Security⇨Trusted Sources).

В этой главе рассматривается создание кода для автоматизации работы редактора VBE, которое будет продемонстрировано на примере разработки инструментария VBE, позволяющего ускорить разработку приложений. После этого, для демонстрации программной модификации кода, диалоговых окон `UserForm` и ссылок, в инструментарий будет добавлено несколько утилит. Для сохранения простоты большая часть кода в этой главе не содержит обработки ошибок.

Идентификация объектов редактора VBE

Все составляющие редактор VBE объекты, свойства и методы хранятся в собственной объектной библиотеке. Перед их использованием в проекте придется создать ссылку на данную библиотеку. Для этого нужно открыть редактор VBE, выбрать меню Tools⇨ References (Сервис⇨Ссылки), установить флажок напротив Microsoft Visual Basic for Applications Extensibility 5.3 library, как показано на рис. 14.1, и щелкнуть на кнопке OK.

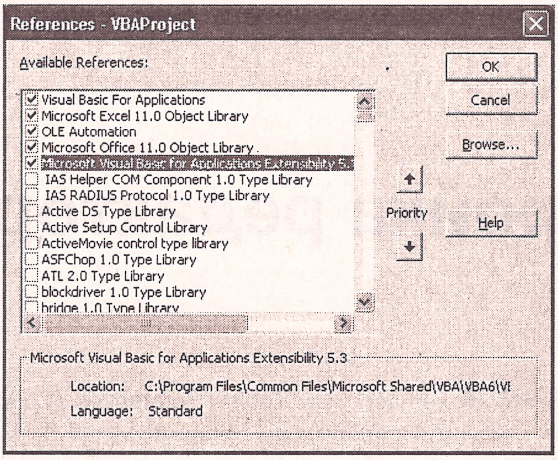


Рис. 14.1. Добавление ссылки на объектную библиотеку редактора VBE

Объект VBE

Объект верхнего уровня в объектной модели редактора называется `VBE`. Этот объект является свойством объекта Excel `Application`. Таким образом, для создания объектной переменной, которая ссылается на объект VBE, можно воспользоваться следующим кодом:

```
Dim VBE As VBIDE.VBE
Set VBE = Application.VBE
```

Объект VBProject

Этот объект является контейнером для всех "программных" компонентов книги, включая диалоговые окна `UserForm`, стандартные модули, модули классов и код каждого листа и книги. Для доступа к объектам `VBProject` можно просмотреть коллекцию `VBProjects` или воспользоваться свойством `VBProject` объекта книги.

Для поиска объекта `VBProject`, соответствующего книге `Book1.xls`, воспользуйтесь следующим кодом:

```
Dim project As VBIDE.VBProject
Set project = Workbooks("Book1.xls").VBProject
```

При создании надстройки для среды разработки VB может потребоваться информация о выделенном в данный момент проекте в Project Explorer (Окне проекта). Для этого воспользуйтесь свойством `ActiveVBProject` объекта `VBE`:

```
Dim project As VBIDE.VBProject
Set project = Application.VBE.ActiveVBProject
```

Обратите внимание, что свойство ActiveVBProject указывает на проект, который редактируется в редакторе VBE. Это свойство никак не связано со свойством ActiveWorkbook, которое предоставляется Excel. На самом деле, с выходом Office 2000 Developer Edition появилась возможность создавать изолированные проекты VB, не являющиеся частью книги Excel.

Объект VBComponent

В модели расширения функциональности диалоговые окна UserForm, стандартные модули, модули классов и модули кода листов и книг представлены объектом VBComponent. Каждый объект VBComponent соответствует одному из компонентов нижнего уровня в Project Explorer (Окне проекта). Для получения доступа к конкретному объекту VBComponent можно воспользоваться коллекцией VBComponents объекта VBProject. Таким образом, следующий код применяется для поиска объекта VBComponent, который представляет диалоговое окно UserForm1 в книге Book1.xls:

```
Dim component As VBIDE.VBComponent
Set component =
↳ Workbooks("Book1.xls").VBProject.VBComponents("UserForm1")
```

Имя объекта VBComponent, в котором хранится код книги, листа или диаграммы, доступно через свойство CodeName соответствующего объекта Excel (книги, листа или диаграммы). Таким образом, следующий код можно использовать для доступа к объекту VBComponent с кодом книги (код в этом объекте может реагировать на события книги):

```
Dim component As VBIDE.VBComponent
Set component = Workbooks("Book1.xls").VBProject.VBComponents( _
    Workbooks("Book1.xls").CodeName)
MsgBox component.Name
```

Для конкретного листа можно применять следующий код:

```
Dim component As VBIDE.VBComponent
Dim aWorkbook As Workbook
Set aWorkbook = Workbooks("Book1.xls")
Set component = aWorkbook.VBProject.VBComponents( _
   aWorkbook.Worksheets("Sheet1").CodeName)
MsgBox component.Name
```

Обратите внимание, что в окне проекта в качестве имени соответствующего книге объекта VBComponent используется ThisWorkbook. Не поддавайтесь соблазну применить это имя в собственном коде. Если пользователь выберет другой язык интерфейса Office, имя будет другим. Кроме этого, пользователь может изменить имя в редакторе VBE. По этой причине не применяйте следующий код:

```
Dim component As VBIDE.VBComponent

With Workbooks("Book1.xls")
    Set component = .VBProject.VBComponents("ThisWorkbook")
End With
```

При разработке надстроек для VBE может потребоваться информация об объекте VBComponent, который редактируется пользователем (выделенном в окне проекта). Ссылка на этот объект доступна в качестве значения свойства SelectedVBComponent объекта редактора VBE:

```
Dim component As VBIDE.VBComponent
Set component = Application.VBE.SelectedVBComponent
```

Каждый объект VBComponent предоставляет коллекцию Properties, содержимое которой примерно соответствует списку в окне Properties (Свойства) при выборе компонента в окне Project Explorer (Окно проекта). Одним из свойств является свойство Name, использование которого показано в следующем коде:

```
Public Sub ShowNames()
    With Application.VBE.SelectedVBComponent
        Debug.Print .Name & ": " & .Properties("Name")
    End With
End Sub
```

Для большинства объектов VBComponent текст свойств .Name и .Properties("Name") совпадает. Но в случае объектов VBComponent, которые хранят код книг, листов и диаграмм, свойство .Properties("Name") возвращает имя объекта Excel (листа, книги или диаграммы). Эту особенность можно использовать для поиска объектов Excel, соответствующих редактируемым элементам в VBE или книге Excel, на которую указывает свойство ActiveVBProject. Соответствующий код показан далее в этой главе.

Объект CodeModule

Весь код VBA для объекта VBComponent хранится в объекте CodeModule. Этот объект позволяет программно читать, добавлять, модифицировать и удалять строки кода. Для каждого объекта VBComponent существует только один объект CodeModule. Каждый тип VBComponent хранит объект CodeModule, но в следующих версиях это может меняться. Например, может быть создан инструмент для проектирования, выполнения и отладки запросов SQL, например Microsoft Query, не содержащий собственного кода.

Объект CodePane

Данный объект предоставляет доступ к пользовательскому представлению содержимого CodeModule. Через этот объект можно идентифицировать такие элементы, как видимая на экране часть CodeModule и последний выделенный текст. Для идентификации редактируемого в данный момент объекта CodePane можно воспользоваться свойством ActiveCodePane редактора VBE:

```
Dim codePane As VBIDE.codePane
Set codePane = Application.VBE.ActiveCodePane
MsgBox codePane.TopLine
```

Например, в результате работы предыдущего кода выделяется верхняя видимая строка из панели кода. Если прокрутить экран, верхней видимой строкой может оказаться 34-я строка модуля.

Объект Designer

Некоторые объекты VBComponent (например, диалоговые окна UserForm) предоставляют разработчику как код, так и графическое представление. Доступ к коду обеспечивается объектами CodeModule и CodePane. Объект Designer предоставляет доступ к графическому представлению объекта. В стандартных версиях Office диалоговые окна UserForm являются единственным компонентами с управляемым графическим

представлением. Но в Developer Edition предоставляется еще целый ряд объектов (например, Data Connection Designer) с графическим представлением. Это объекты также доступны через объект `Designer`.

В оставшейся части главы в основном будут использоваться перечисленные объекты, на основе которых будет создаваться надстройка VBE Toolkit.

Начинаем

В Office 2003 разница между надстройкой книги и надстройкой COM невелика. И для кода, и для `UserForm` предоставляется одинаковый уровень защиты (с сокрытием проекта от просмотра). Основными преимуществами надстроек COM с точки зрения хранения инструментария является возможность сокрытия источника инструментария в пользовательском интерфейсе Excel. Кроме этого, надстройки COM можно загружать через диалоговое окно Сервис⇨Надстройки (Tools⇨Addins) (хотя активизация каждой надстройки немного замедляет процесс загрузки Excel). В этой главе термин "надстройка" используется для обозначения контейнера, содержащего инструментарий, который будет применяться в Excel или VBE. На самом деле в процессе разработки надстройки файл имеет формат стандартной книги. В формат надстройки он будет преобразован только в конце процесса разработки.

Вот базовая структура надстройки:

- базовый модуль перехватывает открытие и закрытие надстройки;
- специальный код добавляет пункты меню при открытии и удаляет пункты меню при закрытии;
- модуль класса обрабатывает события `Click` для пунктов меню;
- наконец, основной код выполняет операции, связанные с пунктами меню.

Для начала откройте новую книгу и удалите все листы, кроме `Sheet1`. Нажмите комбинацию клавиш <Alt+F11> для переключения в редактор VBE. Найдите книгу в окне Project Explorer (Окно проекта). Выберите `VBProject`. В окне Properties (Свойства) измените имя проекта на **`VBETools`**. Добавьте в проект новый модуль и назовите его `Common`. Введите в модуль следующий код. Этот код будет запускаться при каждом открытии и закрытии книги.

```
Option Explicit
Option Compare Text

Public Const AddinID As String = "VBETools"
Public Const Title As String = "VBE Tools"

Sub Auto_Open()
    SetUpMenus
End Sub

Sub Auto_Close()
    RemoveMenus
End Sub
```

Процедуры `Auto_Open` и `Auto_Close` вызывают другие подпрограммы (эти подпрограммы рассматриваются в следующем разделе), которые будут добавлять и удалять меню и пункты меню в редакторе VBE. Кроме этого, определяется глобальная константа, уникально идентифицирующая меню надстройки. Еще одна константа используется в качестве стандартного заголовка надстройки в окнах сообщений.

Добавление пунктов меню в редакторе VBE

В редакторе VBE используется тот же код панелей меню, что и в остальных компонентах пакета Office, поэтому процедура добавления собственных пунктов меню в редактор VBE практически ничем не отличается от примеров, которые будут показаны в главе 26. Но есть одно отличие, заключающееся в способе вызова подпрограммы при щелчке на пункте меню. При добавлении пунктов меню в Excel свойству OnAction объекта CommandBarButton присваивается имя запускаемой подпрограммы. В редакторе VBE объекты CommandBarButton также обладают свойством OnAction, но значение этого свойства игнорируется.

Вместо этого компания Microsoft добавила в объект CommandBarButton событие Click (и событие Change в объект CommandBarComboBox). Для использования этих событий необходимо создать модуль класса, в котором с помощью ключевого слова WithEvents объявлена объектная переменная подходящего типа. Для того чтобы переменную можно было использовать для обработки событий пунктов меню, добавьте в проект модуль класса, назовите его CommandBarEventsClass и добавьте следующую реализацию процедуры обработки события CommandBarEvents_Click:

```
Private WithEvents CommandBarEvents As CommandBarButton

Private Sub CommandBarEvents_Click(ByVal Ctrl As Office.CommandBarButton, _
    CancelDefault As Boolean)
    On Error Resume Next
    Application.Run Ctrl.OnAction
    CancelDefault = True
End Sub
```

Обратите особое внимание на:

- объект CommandBarEvents, который используется для получения события Click от пунктов меню;
- событие Click, создаваемое объектом CommandBarButtonEvents (это единственное событие, которое предоставляется этим объектом);
- объект Ctrl (пункт меню или кнопка на панели инструментов), который передается вместе с событием Click;
- код, запускающий подпрограмму, указанную в качестве значения свойства OnAction. Этот код позволяет эмулировать поведение Excel при добавлении собственных пунктов меню.

Вот пример кода, который объявляет ссылку на класс CommandBarEventsClass. После создания этого класса обработчик события AboutMe связывается с открытой переменной CommandBarEvents.

```
Dim Events As CommandBarEventsClass

Public Sub AddMenu()
    Dim AddinBar As CommandBar
    Dim Button As CommandBarButton

    Set AddinBar =
↳ Application.VBE.CommandBars.FindControl(ID:=30038).CommandBar
    Set Button = AddinBar.Controls.Add(msoControlButton)

    Button.Caption = "О My Addin"
    Button.Tag = "MyAddin"
```

```
    Button.OnAction = "AboutMe"

    Set Events = New CommandBarEventsClass

    Set Events.CommandBarEvents = Button

End Sub

Sub AboutMe()
    MsgBox "О себе"
End Sub
```

Для использования этого класса его необходимо связать с каждым добавленным объектом CommandBarButton. Для этого воспользуйтесь показанным выше кодом, который можно добавить в новый стандартный модуль. При связывании обработчика события с объектом CommandBarButton обработчик события (переменная CommandBarEvents в классе CommandBarEventsClass) на самом деле связывается со свойством Tag объекта пункта меню.

```
Option Explicit
Dim Events As CommandBarEventsClass

Public Sub AddMenu2()

    Dim AddinBar As CommandBar
    Dim Button As CommandBarButton

    Set AddinBar =
➥ Application.VBE.CommandBars.FindControl(ID:=30038).CommandBar
    Set Button = AddinBar.Controls.Add(msoControlButton)

    Button.Caption = "О My Addin"
    Button.Tag = "MyAddin"
    Button.OnAction = "AboutMe"

    Set Events = New CommandBarEventsClass
    Set Events.CommandBarEvents = Button

    Set Button = AddinBar.Controls.Add(msoControlButton)

    Button.Caption = "И о My Addin"
    Button.Tag = "MyAddin"
    Button.OnAction = "AboutMeToo"

End Sub

Sub AboutMe()
  MsgBox "Обо мне"
End Sub

Sub AboutMeToo()
  MsgBox "И обо мне"
End Sub
```

Как показано в этом примере, с добавлением кнопки все кнопки с одинаковым значением свойства Tag будут создавать событие Click в пределах одного экземпляра класса CommandBarEventsClass. Щелчки на обоих кнопках обрабатываются единственным объектом Events. Обратите внимание, что предыдущий код не является частью надстройки Инструментария VBE, поэтому модуль стоит удалить.

Создание меню на основе таблиц

Профессиональные разработчики приложений Excel редко добавляют собственные пункты меню по одному. Большинство разработчиков используют для этих целей табличный подход, при котором в таблицу заносится вся информация о пунктах меню и подпрограмма генерирует их на основе этой информации. Ниже показано использование данного приема. Подход на основе таблиц предоставляет несколько преимуществ:

- один и тот же код создания меню может использоваться в различных проектах;
- намного проще добавить строки в таблицу, чем модифицировать код;
- структуру меню проще определить по содержимому таблицы, чем по эквивалентному коду.

Сначала необходимо создать таблицу, в которой будет храниться информация о меню. В Excel переименуйте лист в `MenuTable` и введите данные, показанные на рис. 14.2.

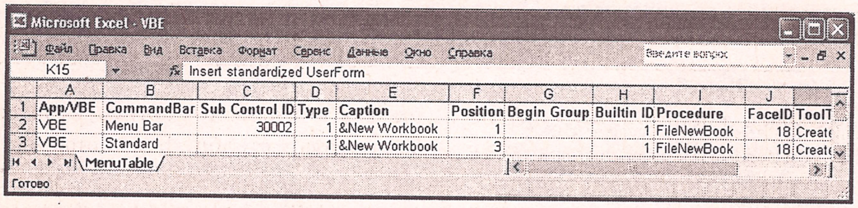


Рис. 14.2. Описание структуры меню

На листе `MenuTable` присутствуют следующие столбцы:

Столбец	*Название*	*Описание*
A	App / VBE	"App" для добавления пунктов меню в Excel или "VBE" для добавления пунктов меню в редактор VBE
B	CommandBar	Имя панели инструментов верхнего уровня, в которую будет добавлено меню. Далее показан список действительных имен и компонентов редактора VBE, которые можно указывать в этом столбце
C	Sub Control ID	Идентификатор встроенного всплывающего меню, в которое добавляются пункты меню. Например, 30002 является идентификатором меню Файл (File)
D	Type	Тип добавляемого элемента управления: 1 — нормальная кнопка, 10 — всплывающее меню и т.д. Значение этого поля соответствует значениям типов `msoControl`, перечисленным в окне Object Browser (Просмотр объектов)
E	Caption	Текст пункта меню
F	Position	Положение добавляемого пункта меню на панели инструментов. Оставьте это поле пустым для добавления пункта в конец панели инструментов
G	Begin Group	True или False для указания разделителя перед пунктом меню

Столбец	*Название*	*Описание*
H	BuiltIn ID	При добавлении пункта во встроенное меню в этом поле указывается идентификатор меню. Для пунктов собственных меню используется идентификатор 1
I	Procedure	Имя процедуры, которая вызывается при щелчке на пункте меню
J	FaceID	Идентификатор встроенной пиктограммы, используемой для обозначения пункта меню. Кроме этого, в качестве значения этого поля можно указывать имя изображения на листе, которое будет использоваться в качестве пиктограммы на кнопке. На кнопке Создать (New) используется пиктограмма с идентификатором 18
K	ToolTip	Текст всплывающей подсказки
L	Popup1 - n	При добавлении собственных всплывающих меню в этом поле указывается название меню, в которое будут добавляться собственные пункты. Использование этого параметра показано далее в этой главе. Можно создавать всплывающие меню любой глубины вложенности. Для этого достаточно добавить столбец и код автоматически обработает увеличение размерности исходных данных

В следующей таблице представлены имена каждой панели инструментов верхнего уровня редактора VBE (эти имена можно использовать в столбце B таблицы меню). Обратите внимание, что Excel распознает эти имена вне зависимости от языка пользовательского интерфейса Office (за редкими исключениями, например, при использовании меню на голландском языке; в этом случае будет выдано сообщение об ошибке времени выполнения). Но к добавляемым пунктам это не относится. Единственным независимым от языка способом поиска конкретного пункта меню является использование номера идентификатора. Подпрограмма для перечисления идентификаторов пунктов встроенных меню рассматривается в главе 26.

Имя	*Описание*
Menu Bar	Стандартное меню VBE
Standard	Стандартная панель инструментов VBE
Edit	Панель инструментов редактирования, на которой содержатся полезные инструменты редактирования кода
Debug	Панель отладочных инструментов
UserForm	Панель инструментов UserForm, на которой содержатся инструменты для редактирования диалоговых окон
MSForms	Всплывающее меню для диалоговых окон `UserForm` (отображается при щелчке правой кнопкой мыши на фоне диалогового окна `UserForm`)
MSForms Control	Всплывающее меню для нормального элемента управления на диалоговом окне `UserForm`
MSForms Control Group	Всплывающее меню, которое отображается при щелчке правой кнопкой мыши на группе элементов управления в диалоговом окне `UserForm`

Имя	*Описание*
MSForms MPC	Всплывающее меню для элемента управления `Multipage`
MSForms Palette	Всплывающее меню, отображающееся при щелчке правой кнопкой мыши в окне Control Toolbox
MSForms DragDrop	Всплывающее меню, которое отображается при перетаскивании элемента управления между вкладками диалогового окна Control Toolbox или на диалоговое окно `UserForm` с помощью правой кнопки мыши
Code Window	Всплывающее меню для окна кода
Code Window (Break)	Всплывающее меню для окна кода в режиме Break (отладка)
Watch Window	Всплывающее меню для окна Watch
Immediate Window	Всплывающее меню для окна Immediate
Locals Window	Всплывающее меню для окна Locals
Project Window	Всплывающее меню для окна Project Explorer
Project Explorer (Break)	Всплывающее меню для окна Project Explorer в режиме Break
Object Browser	Всплывающее меню для окна Object Browser
Property Browser	Всплывающее меню для окна Properties
Docked Window	Всплывающее меню, которое отображается при щелчке правой кнопкой мыши на заголовке присоединенного окна
Project Window Insert	Всплывающее меню, позволяющее добавить в книгу новое диалоговое окно `UserForm`, модуль или класс
Document	Всплывающее меню, позволяющее сохранять книгу, импортировать файл или запускать печать документа
Toggle	Всплывающее меню, позволяющее переключать состояние закладки или точки останова
Toolbox	Всплывающее меню, позволяющее добавлять вкладку в окно инструментов, присоединять и отсоединять окно, а также скрывать окно инструментов
Toolbox Group	Всплывающее меню, предоставляющее возможность добавления, удаления и переименования вкладок в окнах инструментов, а также перемещения окон инструментов вверх и вниз
Task Pane	Всплывающее меню для панели инструментов
Clipboard	Всплывающее меню для буфера обмена
Envelope	Всплывающее меню для конверта
System	Всплывающее системное меню, в котором предоставляется возможность сворачивания, разворачивания, перемещения, закрытия и изменения размера окна
Online Meeting	Всплывающее меню для интерактивного взаимодействия пользователей

Так как на этот лист придется неоднократно ссылаться из кода, желательно присвоить ему подходящее имя, например MenuTable. Для этого найдите и выделите лист в окне Project Explorer (Окно проекта) в редакторе VBE. Скорее всего, лист называется Sheet1 (MenuTable). Измените имя листа в окне Properties (Свойства). После этого в окне Project Explorer (Окно проекта) лист будет называться MenuTable (MenuTable). Переименование позволяет ссылаться на лист как на объект, что делает следующие две строки кода эквивалентными:

```
Debug.Print ThisWorkbook.Worksheets("MenuTable").Name
Debug.Print MenuTable.Name
```

Ниже показан код для создания меню из таблицы. Этот код необходимо скопировать в модуль SetupCommandBars.

В начале модуля объявляется несколько констант, соответствующих столбцам в таблице меню. Эти константы будут использоваться в пределах всего кода. При изменении структуры меню достаточно будет перенумеровать константы, что избавит от поиска по всему коду.

```
Option Explicit
Option Compare Text

Const TABLE_APP_VBE           As Integer = 1
Const TABLE_COMMANDBAR_NAME   As Integer = 2
Const TABLE_CONTROL_ID        As Integer = 3
Const TABLE_CONTROL_TYPE      As Integer = 4
Const TABLE_CONTROL_CAPTION   As Integer = 5
Const TABLE_CONTROL_POSITION As Integer = 6
Const TABLE_CONTROL_GROUP     As Integer = 7
Const TABLE_CONTROL_BUILTIN As Integer = 8
Const TABLE_CONTROL_PROC     As Integer = 9
Const TABLE_CONTROL_FACEID   As Integer = 10
Const TABLE_CONTROL_TOOLTIP As Integer = 11
Const TABLE_POPUP_START      As Integer = 12

Dim Events As CommandBarEventsClass
```

Как было показано выше, событие Click для всех панелей может направляться на единственный экземпляр обработчика событий. Для этого все объекты должны иметь одно и то же значение свойства Tag.

```
Public Sub SetUpMenus()
    Dim aRange As Range
    Dim items As CommandBars
    Dim item As CommandBar
    Dim control As CommandBarControl
    Dim builtInId As Integer
    Dim column As Integer
    Dim data As Variant

    On Error Goto Catch

    RemoveMenus

    For Each aRange In MenuTable.Cells(1).CurrentRegion.Rows
        If aRange.row > 1 Then
            data = aRange.Value
            Set item = Nothing
```

Подпрограмма для фактической настройки меню вызывается из процедуры Auto_Open:

```
If data(1, TABLE_APP_VBE) = "VBE" Then
  Set items = Application.VBE.CommandBars
Else
  Set items = Application.CommandBars
End If

Set item = items.item(data(1, TABLE_COMMANDBAR_NAME))

If item Is Nothing Then
    Set item = items.Add(name:=data(1, TABLE_COMMANDBAR_NAME), _
        temporary:=True)
End If
```

Одна процедура может использоваться для добавления пунктов в меню Excel и VBE. Единственным отличием является выбор коллекции CommandBars из Excel или коллекции CommandBars из VBE:

```
If Not IsEmpty(data(1, TABLE_CONTROL_ID)) Then
    Set item = item.FindControl(ID:=data(1, TABLE_CONTROL_ID), _
        Recursive:=True).CommandBar
End If

For column = TABLE_POPUP_START To UBound(data, 2)
    If Not IsEmpty(data(1, column)) Then
        Set item = item.Controls(data(1, column)).CommandBar
    End If
Next

builtInId = data(1, TABLE_CONTROL_BUILTIN)

If builtInId = 0 Then builtInId = 1

If IsEmpty(data(1, TABLE_CONTROL_POSITION)) Or _
    data(1, TABLE_CONTROL_POSITION) > item.Controls.Count Then

    Set control = item.Controls.Add(Type:=data(1,
↳ TABLE_CONTROL_TYPE), _
        ID:=builtInId, temporary:=True)
Else
    Set control = item.Controls.Add(Type:=data(1,
↳ TABLE_CONTROL_TYPE), _
        ID:=builtInId, temporary:=True, before:=data(1,
↳ TABLE_CONTROL_POSITION))
End If

control.Caption = data(1, TABLE_CONTROL_CAPTION)
control.BeginGroup = data(1, TABLE_CONTROL_GROUP)
control.TooltipText = data(1, TABLE_CONTROL_TOOLTIP)
```

Если элемент управления необходимо добавить во встроенное всплывающее меню, можно воспользоваться рекурсивным поиском в коллекции CommandBars. Например, если необходимо добавить пункт в меню Format⇨Make same size, идентификатор меню Make same size (32790) можно указать в столбце SubControlID таблицы:

```
If Not IsEmpty(data(1, TABLE_CONTROL_FACEID)) Then
    If IsNumeric(data(1, TABLE_CONTROL_FACEID)) Then
        control.FaceId = data(1, TABLE_CONTROL_FACEID)
    Else
        MenuTable.Shapes(data(1, TABLE_CONTROL_FACEID)).CopyPicture
```

```
            control.PasteFace
        End If
    End If

    control.Tag = AddinID
    control.OnAction = "'" & ThisWorkbook.name & "'!" & _
        data(1, TABLE_CONTROL_PROC)
```

Можно использовать стандартную пиктограмму Office (указав числовое значение `FaceID`) или воспользоваться собственной пиктограммой. Для применения собственной пиктограммы в столбце `FaceID` в таблице меню укажите имя объекта `Picture`. Изображение должно иметь размеры 32 x 32 для маленькой пиктограммы и 64 × 64 для большой пиктограммы:

```
        If builtInId = 1 And data(1, TABLE_APP_VBE) = "VBE" Then
            If Events Is Nothing Then
                Set Events = New CommandBarEventsClass
                Set Events.CommandBarEvents = control
            End If

        End If
    End If
    Next
    Exit Sub
    Catch:
    MsgBox Err.Description
    End Sub
```

(Если выдается сообщение об ошибке "Программный доступ к проекту Visual Basic не является доверенным", выберите в Excel пункт Сервис⇨Параметры⇨Безопасность (Tools⇨Options⇨Security), активизируйте вкладку Надежные издатели (Trusted Publishers) и установите флажок Доверять доступ к Visual Basic Project (Trusted access to Visual Basic Projects).) Этот код создает обработчик событий меню. Обработчик создается один раз, так как он обрабатывает события `Click` для всех кнопок.

При закрытии надстройки необходимо запустить код, удаляющий пункты из меню редактора VBE. Некоторые разработчики используют вызов `CommandBars.Reset`, но в результате будут удалены все модификации, а не только модификации, связанные с этой надстройкой. Более правильным является удаление только собственных пунктов меню и панелей. Для этого потребуются две подпрограммы. Первая удаляет пункты меню из конкретной коллекции `CommandBars`, выполняя поиск по значению свойства `Tag`:

```
Private Sub RemoveMenusFromBars(ByVal items As CommandBars)
    Dim control As CommandBarControl
        On Error Resume Next
    Set control = items.FindControl(Tag:=AddinID)

    Do Until control Is Nothing
        control.Delete
        Set control = items.FindControl(Tag:=AddinID)
    Loop
End Sub
```

Вторая подпрограмма вызывает первую для удаления пунктов меню из панелей Excel и VBE. Кроме этого, вторая подпрограмма удаляет созданные панели и обработчики событий уровня модуля:

```
Public Sub RemoveMenus()
    Dim aCommandBar As CommandBar
    Dim aRange As Range
    Dim name As String

    On Error Resume Next

    Call RemoveMenusFromBars(Application.CommandBars)
    Call RemoveMenusFromBars(Application.VBE.CommandBars)

    For Each aRange In MenuTable.Cells(1).CurrentRegion.Rows

        If aRange.row > 1 Then
        name = aRange.Cells(1, TABLE_COMMANDBAR_NAME)
        Set aCommandBar = Nothing
        If aRange.Cells(1, TABLE_APP_VBE) = "VBE" Then
            Set aCommandBar = Application.VBE.CommandBars(name)
        Else
            Set aCommandBar = Application.CommandBars(name)
        End If

        If Not aCommandBar Is Nothing Then
            If Not aCommandBar.BuiltIn Then
                If aCommandBar.Controls.Count = 0 Then
⇘ aCommandBar.Delete
                End If
            End If
        End If
    Next

    Set Events = Nothing
End Sub
```

На данный момент создан полный шаблон, который можно использовать в качестве основы для любой надстройки Excel и VBE (или в качестве стандартной книги, позволяющей модифицировать структуру меню).

Добавим следующую процедуру и проверим работу надстройки. Создайте новый модуль `MenuFile` и скопируйте в него следующий код. Дополнительные, связанные с файлами, подпрограммы будут добавлены позднее:

```
Public Sub FileNewWorkbook()
    On Error Resume Next
    Application.Workbooks.Add

    Application.VBE.MainWindow.Visible = False
    Application.VBE.MainWindow.Visible = True
End Sub
```

Метод `FileNewWorkbook` просто создает пустую книгу и обновляет содержимое окна VBE. Обратите внимание, что при добавлении или удалении книг из кода окно VBE Project Explorer не всегда обновляется корректно. Самым простым способом обновления содержимого окна VBE является сокрытие и повторное отображение.

Сначала убедитесь в правильной компиляции надстройки. Для этого выберите команду Debug⇨Compile (Отладка⇨Компиляция). Если в процессе компиляции были обнаруже-

ны ошибки, сравните код с показанными выше листингами. Для запуска надстройки выберите пункт Tools⇨Macros (Сервис⇨Макрос), выделите подпрограмму `Auto_Open` и щелкните на кнопке Run (Выполнить). В результате правильного выполнения кода в меню File (Файл) будет добавлен новый пункт и на панель инструментов VBE слева от пиктограммы Save (Сохранить) будет добавлена пиктограмма New (Создать) (рис. 14.3 и рис. 14.4).

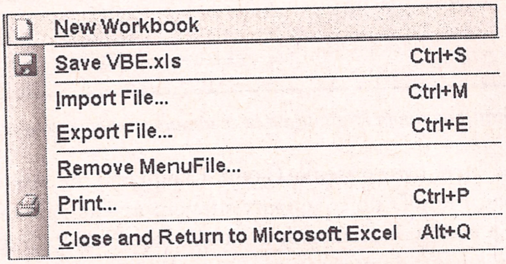


Рис. 14.3. Новый пункт меню

Рис. 14.4. Новая кнопка

При щелчке на кнопке в Excel создается новая книга и в окно Project Explorer (Окно проекта) добавляется новый объект `VBProject`. Поздравляем, вы успешно модифицировали VBE!

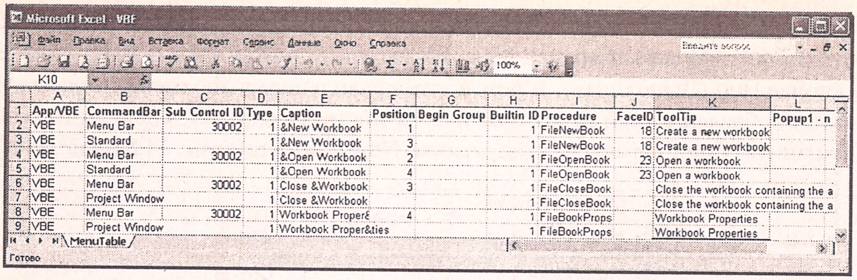
Вывод встроенных диалоговых окон, диалоговых окон UserForm и окон сообщений

В пакете Office 2003 предоставляется возможность сохранения книги из VBE, но ранее была добавлена возможность создания новой книги. Для полного набора файловых операций необходимо добавить подпрограммы для открытия и закрытия книг. Добавление списка последних открывавшихся книг остается читателю в качестве самостоятельного упражнения.

К завершению этой главы в VBE будет добавлена следующая функциональность:

- создание новой книги;
- открытие существующей книги;
- сохранение книги (эта возможность встроена в редактор VBE);
- закрытие книги;
- вывод диалогового окна Properties (Свойства) со свойствами книги.

Для подпрограммы `Open` добавляется один пункт в меню File (Файл) и одна кнопка на панель инструментов. Для подпрограммы `Close` добавляется еще один пункт в меню File (Файл) и во всплывающее меню Project Explorer (Окно проекта). В результате, щелкнув правой кнопкой мыши на окне Project Explorer (Окно проекта), можно будет закрыть объект `VBProject`. На рис. 14.5 показаны необходимые дополнения к таблице меню.



App/VBE	CommandBar	Sub Control ID	Type	Caption	Position	Begin Group	Builtin ID	Procedure	FaceID	ToolTip	Popup1 - n
VBE	Menu Bar	30002	1	&New Workbook	1		1	FileNewBook	18	Create a new workbook	
VBE	Standard		1	&New Workbook	3		1	FileNewBook	18	Create a new workbook	
VBE	Menu Bar	30002	1	&Open Workbook	2		1	FileOpenBook	23	Open a workbook	
VBE	Standard		1	&Open Workbook	4		1	FileOpenBook	23	Open a workbook	
VBE	Menu Bar	30002	1	Close &Workbook	3		1	FileCloseBook		Close the workbook containing the a	
VBE	Project Window		1	Close &Workbook			1	FileCloseBook		Close the workbook containing the a	
VBE	Menu Bar	30002	1	Workbook Proper&	4		1	FileBookProps		Workbook Properties	
VBE	Project Window		1	Workbook Proper&ties			1	FileBookProps		Workbook Properties	

Рис. 14.5. Модифицированная таблица для создания интересующих пунктов меню

Обратите внимание, что пункт меню Close (Закрыть) не имеет стандартной пиктограммы, поэтому столбец FaceID не содержит значения. Так как позиция пункта меню не указана, он добавляется в конец всплывающего меню окна Project Explorer (Окно проекта).

Для правильной эмуляции функциональности Excel необходимо проверить состояние клавиши <Shift> в момент щелчка на пункте меню. Если в момент щелчка она удерживается, необходимо отключить обработку всех событий. Удерживание клавиши <Shift> при открытой книге приводит к удалению подпрограммы (дополнительную информацию можно найти в статье Microsoft Knowledge Base Q175223 по адресу `http://support.microsoft.com/support/kb/articles/Q175/2/23.asp`), но в Excel 2003 эта проблема уже решена. Лучшее, что можно сделать, это добавить клавишу <Ctrl> для достижения того же эффекта. Добавьте следующее объявление в начало модуля `Common`:

```
Private Declare Function GetAsyncKeyState Lib "user32"
↳ (ByVal vKey As Long) As Integer
```

Это объявление сообщает VBA о функции, доступной в Windows API. Дополнительная информация о вызове методов Windows API приводится в главе 16. Добавьте следующую функцию в конец модуля `Common`:

```
Public Function GetShiftCtrlAlt() As Integer
    Dim Keys As Integer
    Const VK_SHIFT As Long = &H10
    Const VK_CONTROL As Long = &H11
    Const VK_ALT As Long = &H12

    ' Проверить состояние клавиш <Ctrl>, <Shift> и <Alt>
    If GetAsyncKeyState(VK_SHIFT) <> 0 Then Keys = Keys + 1
    If GetAsyncKeyState(VK_CONTROL) <> 0 Then Keys = Keys + 2
    If GetAsyncKeyState(VK_ALT) <> 0 Then Keys = Keys + 4

    GetShiftCtrlAlt = Keys
End Function
```

В подпрограмме `Open` используется метод Excel `GetOpenFilename`, который позволяет найти имя файла и открыть файл. Если пользователь удерживает клавишу <Ctrl>, события приложения будут отключены, поэтому пользователь может открыть книгу, не запуская код в открывающейся книге или в собственной подпрограмме Excel `Workbook_Open`. Если пользователь не удерживает клавишу <Ctrl>, то делается попытка запустить все подпрограммы `Auto_Open` в пределах книги:

```
Public Sub FileOpenBook()
    Dim FileName As Variant
    Dim IsControlKeyPressed As Boolean
    Dim aWorkbook As Workbook

    On Error GoTo Catch

    IsControlKeyPressed = (GetShiftCtrlAlt And 2) = 2
    Application.Visible = False
    FileName = Application.GetOpenFilename
    Application.Visible = True

    If Not (FileName = False) Then
        If IsControlKeyPressed Then
            Application.EnableEvents = False

            Set aWorkbook = Workbooks.Open(FileName:=FileName, _
            UpdateLinks:=0, AddToMru:=True)
            Application.EnableEvents = True
        Else
            Set aWorkbook = Workbooks.Open(FileName:=FileName, _
            AddToMru:=True)
            aWorkbook.RunAutoMacros xlAutoOpen
        End If
    End If

    Application.VBE.MainWindow.Visible = False
    Application.VBE.MainWindow.Visible = True
    Exit Sub

Catch:
    Application.Visible = False
    MsgBox Err.Description, vbCritical
    Application.Visible = True
End Sub
```

При каждом использовании диалогового окна, которое должно отображаться в пределах окна Excel (включая встроенные диалоговые окна, диалоговые окна `UserForm` и даже диалоговые окна функций `MsgBox` и `InputBox`), Excel автоматически переключает собственное окно на отображение диалогового окна. Но при разработке приложений для редактора VBE диалоговое окно должно отображаться внутри окна VBE, а не в окне Excel. Самый простой способ получения такого эффекта – это сокрытие окна Excel перед отображением собственного диалогового окна.

```
Public Function ActiveProjectBook() As Workbook
    Dim project As VBIDE.VBProject
    Dim component As VBIDE.VBComponent
    Dim name As String

    On Error GoTo Finally

    Set project = Application.VBE.ActiveVBProject

    If project.Protection = vbext_pp_locked Then
        name = project.FileName

        If InStrRev(name, "\") <> 0 Then
            name = Mid(name, InStrRev(name, "\") + 1)
                If IsWorkbook(name) Then
                    Set ActiveProjectBook = Workbooks(name)
```

```
                Exit Function
            End If
        End If
    Else
    For Each component In project.VBComponents
        If component.Type = vbext_ct_Document Then
            name = component.Properties("Name")
            If IsWorkbook(name) Then
                If Workbooks(name).VBProject Is project Then
                    Set ActiveProjectBook = Workbooks(name)
                    Exit Function
                End If

            End If
        End If
        Next
    End If

Finally:
    Application.Visible = False
    MsgBox "Книга не найдена", vbExclamation
    Application.Visible = True

    Set ActiveProjectBook = Nothing
End Function
```

Подпрограмма `Close` оказывается следующим препятствием. Во всплывающее меню окна Project Explorer (Окно проекта) добавляется пункт Close Workbook (Закрыть книгу), поэтому нужно определить, на каком пункте VBProject щелкнул пользователь. Свойство `ActiveVBProject` редактора VBE содержит интересующую информацию, но необходимо перейти от объекта `VBProject` к книге, в которой содержится данный объект. Этот метод рассматривался в разделе "Идентификация объектов VBE в коде" в начале этой главы. Код этого метода показан ниже. Добавьте следующий код в модуль Common вместе с подпрограммой `Auto_Open` и завершающей функцией.

Подпрограммы `Auto_Close`, которые будут использоваться далее:

```
Public Function IsWorkbook(ByVal book As String) As Boolean

    On Error GoTo Catch
    IsWorkbook = Workbooks(book).name <> ""
    Exit Function

Catch:
    IsWorkbook = False
```

Обратите внимание, что перед отображением сообщения об ошибке окно Excel скрывается и "открывается" после этого. Следующая процедура сравнивает результат с именем книги:

```
Public Sub FileCloseBook()
    Dim aWorkbook As Workbook
    Dim isCtrlKeyPress As Boolean

    On Error GoTo Catch

    Set aWorkbook = ActiveProjectBook
    If aWorkbook Is Nothing Then Exit Sub

    isCtrlKeyPress = (GetShiftCtrlAlt And 2) = 2
```

```
    If isCtrlKeyPress Then
        Application.EnableEvents = False
        aWorkbook.Close
        Application.EnableEvents = True
    Else
        aWorkbook.RunAutoMacros xlAutoClose
        aWorkbook.Close
    End If
    Exit Sub

Catch:
    Application.Visible = False
    MsgBox Err.Description, vbCritical
    Application.Visible = True
End Sub
```

Теперь, когда можно получить доступ к книге, соответствующей активному объекту VBProject, воспользуйтесь подпрограммой Close, которая добавляется в модуль MenuFile:

```
Public Sub FileBookProps()
    Dim aWorkbook As Workbook
    Dim IsAddin As Boolean
    Dim IsVisible As Boolean

    On Error Resume Next
    Set aWorkbook = ActiveProjectBook
    If aWorkbook Is Nothing Then Exit Sub

    Application.Visible = False
    IsAddin = aWorkbook.IsAddin
    aWorkbook.IsAddin = False

    IsVisible = aWorkbook.Windows(1).Visible
    aWorkbook.Windows(1).Visible = True
    Application.Dialogs(xlDialogProperties).Show
    aWorkbook.Windows(1).Visible = IsVisible
    aWorkbook.IsAddin = IsAddin
    Application.Visible = True
End Sub
```

Осталось определить еще один инструмент, связанный с книгой. Он будет использоваться для отображения диалогового окна Свойства файла (File Properties) для книги активного проекта Visual Basic. Одним из основных применений свойств книги является предоставление информации в диалоговом окне Сервис⇨Надстройки (Tools⇨Addins). В списке показан заголовок надстройки из диалогового окна Свойства (Properties). Описание для выделенной надстройки извлекается из поля Комментарий (Comments).

Для этого можно воспользоваться встроенным диалоговым окном Свойства (Properties) из Excel, но при этом сложно определить, для какой книги отображать свойства — используется активная книга. Следовательно, любая надстройка должна быть временно преобразована в нормальную книгу и отображаться, если книга скрыта. После отображения диалогового окна Свойства (Properties) книгу необходимо преобразовать обратно в надстройку.

Для проверки созданной надстройки запустите подпрограмму Auto_Open с помощью команды Сервис⇨Макросы (Tools⇨Macros). В результате выполнения подпрограммы будут созданы все интересующие пункты меню. После этого убедитесь, что все пункты работают правильно.

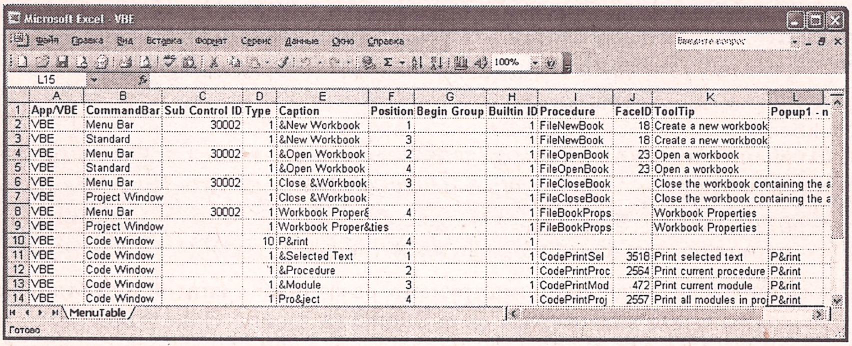
> Обратите внимание, что попытка закрыть надстройку с помощью соответствующего пункта меню может привести к зависанию компьютера.

Работа с кодом

До этого момента рассматривались верхние уровни объектной модели Excel и среды разработки VBE (работа велась в пределах объектов `VBProject` и `Workbook`) для добавления типичных операций с файлами в среде Visual Basic. На данный момент появилась возможность создания новых книг (а значит и проектов Visual Basic), открытия существующих книг, изменения свойств книг, а также сохранения и закрытия книг из интерфейса VBE.

В этом разделе будут рассмотрены нижние уровни объектной модели VBE, а также работа с пользовательским кодом. В данном случае ограничимся определением редактируемой строки (и даже определением выделенного символа в пределах этой строки), а также получением информации о процедуре, модуле и проекте, в которых хранится эта строка кода. Добавление и модификация кода будут рассматриваться в следующем разделе вместе с созданием диалоговых окон `UserForm`, добавлением кнопок и кода для обработки событий кнопок.

Для демонстрации определения редактируемого кода щелчок правой кнопкой мыши позволит получить доступ к подпрограмме выдачи сообщения. Отдельные кнопки позволят выводить текущий выделенный код, название текущей процедуры, модуля или проекта. Для начала добавьте в таблицу меню дополнительные строки (рис. 14.6).



	App/VBE	CommandBar	Sub Control ID	Type	Caption	Position	Begin Group	Builtin ID	Procedure	FaceID	ToolTip	Popup1 - n
2	VBE	Menu Bar	30002	1	&New Workbook	1		1	FileNewBook	18	Create a new workbook	
3	VBE	Standard		1	&New Workbook	3		1	FileNewBook	18	Create a new workbook	
4	VBE	Menu Bar	30002	1	&Open Workbook	2		1	FileOpenBook	23	Open a workbook	
5	VBE	Standard		1	&Open Workbook	4		1	FileOpenBook	23	Open a workbook	
6	VBE	Menu Bar	30002	1	Close &Workbook	3		1	FileCloseBook		Close the workbook containing the a	
7	VBE	Project Window		1	Close &Workbook			1	FileCloseBook		Close the workbook containing the a	
8	VBE	Menu Bar	30002	1	Workbook Proper&	4		1	FileBookProps		Workbook Properties	
9	VBE	Project Window		1	Workbook Proper&ties			1	FileBookProps		Workbook Properties	
10	VBE	Code Window		10	P&rint	4		1				
11	VBE	Code Window		1	&Selected Text	1		1	CodePrintSel	3518	Print selected text	P&rint
12	VBE	Code Window		1	&Procedure	2		1	CodePrintProc	2564	Print current procedure	P&rint
13	VBE	Code Window		1	&Module	3		1	CodePrintMod	472	Print current module	P&rint
14	VBE	Code Window		1	Pro&ject	4		1	CodePrintProj	2557	Print all modules in proj	P&rint

Рис. 14.6. Новые строки таблицы меню

Обратите внимание, что собственное всплывающее меню добавляется во всплывающее меню Code Window (тип 10 соответствует собственному всплывающему меню), после чего в меню добавляются четыре пункта. Каждый пункт имеет собственный идентификатор пиктограммы. Результат показан на рис. 14.7.

Код четырех подпрограмм будет храниться в собственном модуле, поэтому в проект необходимо добавить модуль `MenuCode`.

К сожалению, объектная модель VBE не предоставляет метод `Print` ни для одного объекта. Существует три варианта реализации печати по щелчку правой кнопкой мыши.

- ❑ Вывести диалоговое окно Печать (Print) редактора VBE и управлять им через функцию `SendKeys`.

- Скопировать код в диапазон листа и распечатать диапазон.
- Скопировать код в собственный экземпляр Word, выполнить форматирование для выделения ключевых слов Excel и распечатать код средствами текстового процессора Word.

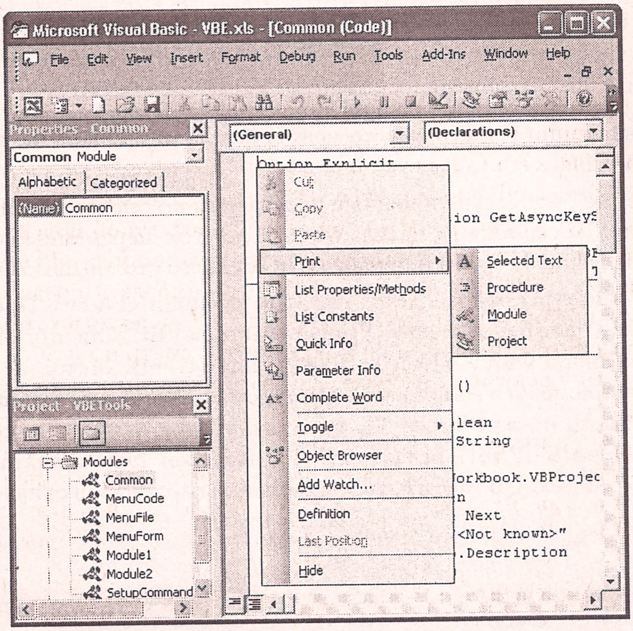


Рис. 14.7. Всплывающее меню, создаваемое в результате работы собственного кода

Для сохранения простоты лучше реализовать первый вариант. Основная проблема заключается в механизме выбора переключателя Selected Text (Выделенный текст), Module (Модуль), Проект (Project) средствами функции SendKeys. Особенно если учитывать, что переключатель Выделенный текст (Selected Text) доступен только при наличии выделенного текста.

Для решения этой проблемы необходимо определить, выделен ли текст, после чего отправить соответствующее количество нажатий клавиши <вниз> для выбора переключателей Модуль (Module) или Проект (Project). Если есть уверенность, что пользователи всегда работают с интерфейсом на английском языке, можно было бы отправить нажатия комбинаций клавиш <Alt+M> или <Alt+J> — отправка нажатия клавиши <вниз> работает с любым языком интерфейса.

Код для выбора переключателя Выделенный текст (Selected Text) является самым простым. В данном случае необходимо определить, существует ли выделение текста, и отправить некоторые нажатия клавиш в диалоговое окно Печать (Print).

```
Option Explicit

Public Sub CodePrintSel()
    Dim StartLine As Long
    Dim StartColumn As Long
    Dim EndLine As Long
    Dim EndColumn As Long
```

```
    Application.VBE.ActiveCodePane.GetSelection StartLine, _
        StartColumn, EndLine, EndColumn

    If StartLine <> EndLine Or StartColumn <> EndColumn Then
        Application.SendKeys "{ENTER}"
        Application.VBE.CommandBars.FindControl(ID:=4).Execute
    End If
End Sub
```

Обратите внимание на следующие моменты:

- для идентификации редактируемого модуля используется свойство ActiveCodePane редактора VBE;
- в метод GetSelection переменные передаются по ссылке и там им присваиваются значения. После вызова метода GetSelection переменные содержат номера первой и последней строки и первого и последнего столбца выделенного текста;
- в буфер клавиатуры отправляется нажатие клавиши <Enter>. После этого отображается диалоговое окно Печать (Print) редактора VBE. Для этого непосредственно выбирается меню Файл⇨Печать (File⇨Print) (ID=4). Метод непосредственного выбора пунктов меню в комбинации с методом RunMenu рассматривается в главе 17. Метод RunMenu показан ниже. По умолчанию (при выделении текста) в диалоговом окне Печать (Print) редактора VBE выделен переключатель Выделенный текст (Selected Text). В таком случае изменение параметров окна не требуется:

```
Public Sub CodePrintMod()
    Dim StartLine As Long
    Dim StartColumn As Long
    Dim EndLine As Long
    Dim EndColumn As Long

    Application.VBE.ActiveCodePane.GetSelection StartLine, _
        StartColumn, EndLine, EndColumn

    If StartLine <> EndLine Or StartColumn <> EndColumn Then
        Application.SendKeys "{DOWN}{ENTER}"
    Else
        Application.SendKeys "{ENTER}"
    End If

    Application.VBE.CommandBars.FindControl(ID:=4).Execute
End Sub

Public Sub CodePrintProj()
    Dim StartLine As Long
    Dim StartColumn As Long
    Dim EndLine As Long
    Dim EndColumn As Long

    Application.VBE.ActiveCodePane.GetSelection StartLine, _
        StartColumn, EndLine, EndColumn
    If StartLine <> EndLine Or StartColumn <> EndColumn Then
        Application.SendKeys "{DOWN}{DOWN}{ENTER}"
    Else
        Application.SendKeys "{DOWN}{ENTER}"
    End If

    Application.VBE.CommandBars.FindControl(ID:=4).Execute
End Sub
```

Для печати текущего модуля и проекта можно использовать очень похожий код. Единственным отличием является необходимость проверки, выделил ли пользователь текст (если это так, в диалоговом окне Печать (Print) будет выбран переключатель Выделенный текст (Selected Text), и отправить необходимое количество нажатий клавиши <вниз> для выбора подходящего переключателя. Обе подпрограммы можно добавить в модуль MenuCode:

```
Public Sub CodePrintProc()
    Dim StartLine As Long
    Dim StartColumn As Long
    Dim EndLine As Long
    Dim EndColumn As Long
    Dim ProcedureType As Long
    Dim ProcedureName As String
    Dim ProcedureStart As Long
    Dim ProcedureEnd As Long

    With Application.VBE.ActiveCodePane
        .GetSelection StartLine, StartColumn, EndLine, EndColumn

        With .CodeModule
            If StartLine <= .CountOfDeclarationLines Then
                ProcedureStart = 1
                ProcedureEnd = .CountOfDeclarationLines
            Else
                ProcedureName = .ProcOfLine(StartLine, ProcedureType)
                ProcedureStart = .ProcStartLine(ProcedureName,
↳ ProcedureType)
                ProcedureEnd = ProcedureStart + _
                .ProcCountLines(ProcedureName, ProcedureType)
            End If
        End With

        Call .SetSelection(ProcedureStart, 1, ProcedureEnd, 999)

        Application.SendKeys "{ENTER}"
        Application.VBE.CommandBars.FindControl(ID:=4).Execute
        DoEvents
        Call .SetSelection(StartLine, StartColumn, EndLine, EndColumn)
    End With
End Sub
```

Код для печати текущей процедуры будет выглядеть более сложным, так как в окне Печать (Print) отсутствует переключатель Текущая процедура (Current Procedure). Придется выполнить следующие операции:

- идентифицировать и сохранить текущее выделение;
- идентифицировать процедуру (или строки объявления), содержащую текущее выделение;
- расширить выделение до полного текста процедуры (или всех строк объявления);
- отобразить диалоговое окно Печать (Print) для печати выделенного текста;
- восстановить начальное выделение.

Выполнение этой последовательности операций на некоторых компьютерах иногда приводит к интересной проблеме — последнее действие по восстановлению выделения выполняется еще до отображения диалогового окна Печать (Print). Одной из причин та-

кого поведения является асинхронная печать. Самое простое решение — это добавление оператора DoEvents непосредственно после оператора отображения диалогового окна Печать (Print). Это позволит подпрограмме печати выполнить поставленную перед ней задачу. Кроме этого, операционной системе передается управление для обработки событий, находящихся в очереди.

Обратите внимание, что метод ProcOfLine принимает в качестве параметра первую строку, присваивает значение переменной ProcedureType для идентификации типа процедуры (Sub, Function, Property Let, Property Get или Property Set) и возвращает имя процедуры. Тип и имя процедуры используются для поиска начала процедуры (с использованием значения ProcStartLine) и количества строк в ней (ProcCountLines). После этого строки процедуры выделяются и отправляются на печать.

Работа с диалоговыми окнами UserForm

Показанные здесь примеры кода использовались для расширения возможностей VBE и предоставления дополнительных инструментов разработчику. В этом разделе основное внимание уделяется программному созданию и управлению диалоговыми окнами UserForm, а также добавлению элементов управления и управляющих процедур в модуль кода диалогового окна UserForm. Хотя в показанных здесь примерах расширяется функциональность VBE, этот же код и приемы могут применяться для пользовательских приложений, включая:

- добавление диалоговых окон UserForm в созданные приложением книги;
- изменение размера диалоговых окон UserForm и элементов управления, а также перемещение элементов управления для максимального использования доступного экранного пространства;
- добавление кода для обработки событий созданных приложением диалоговых окон UserForm;
- изменение элементов управления на существующих диалоговых окнах UserForm в ответ на пользовательский ввод;
- создание диалоговых окон UserForm в процессе работы приложения по мере необходимости (например, когда количество и тип элементов управления на диалоговом окне UserForm значительно отличаются друг от друга в зависимости от природы отображаемых данных).

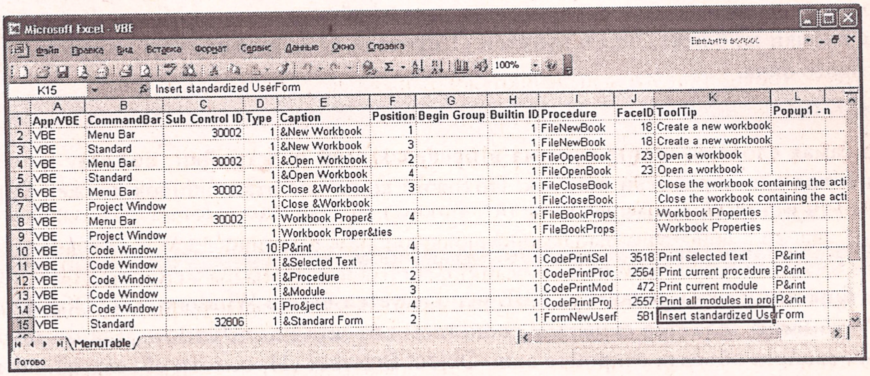
Показанная выше методика реализована в виде кода для добавления диалогового окна UserForm в активный проект. В окне присутствуют стандартные кнопки OK и Отмена (Cancel). Кроме этого, существуют подпрограммы для обработки событий Click для кнопок и события QueryClose для диалогового окна UserForm. Размер диалогового окна UserForm установлен на уровне 2/3 ширины и высоты окна Excel. Положение кнопок OK и Отмена (Cancel) меняется в соответствии с размерами диалогового окна.

В показанном здесь примере используется сложный способ получения интересующего результата. Данный пример, в основном, имеет образовательную, а не практическую ценность. Более простым способом было бы создание диалогового окна вручную с сохранением его на диске в файле .frm. Следующий код позволяет импортировать диалоговое окно из файла (не вводите этот код):

```
Dim component As VBComponent
Set component = Application.VBE.ActiveVBProject. _
    VBComponents.Import ("MyForm.frm")
```

Диалоговое окно легко можно будет включить в другой проект. Единственным преимуществом создания окна с помощью кода является изменение размера и положения в соответствии с разрешением и размером пользовательского экрана.

Начните с добавления еще одной строки в таблицу меню (рис. 14.8).



	App/VBE	CommandBar	Sub Control ID	Type	Caption	Position	Begin Group	Builtin ID	Procedure	FaceID	ToolTip	Popup1 - n
2	VBE	Menu Bar	30002	1	&New Workbook	1		1	FileNewBook	18	Create a new workbook	
3	VBE	Standard		1	&New Workbook	3		1	FileNewBook	18	Create a new workbook	
4	VBE	Menu Bar	30002	1	&Open Workbook	2		1	FileOpenBook	23	Open a workbook	
5	VBE	Standard		1	&Open Workbook	4		1	FileOpenBook	23	Open a workbook	
6	VBE	Menu Bar	30002	1	Close &Workbook	3		1	FileCloseBook		Close the workbook containing the acti	
7	VBE	Project Window		1	Close &Workbook			1	FileCloseBook		Close the workbook containing the acti	
8	VBE	Menu Bar	30002	1	Workbook Proper&	4		1	FileBookProps		Workbook Properties	
9	VBE	Project Window		1	Workbook Proper&ties			1	FileBookProps		Workbook Properties	
10	VBE	Code Window		10	P&rint	4		1				
11	VBE	Code Window		1	&Selected Text	1		1	CodePrintSel	3518	Print selected text	P&rint
12	VBE	Code Window		1	&Procedure	2		1	CodePrintProc	2564	Print current procedure	P&rint
13	VBE	Code Window		1	&Module	3		1	CodePrintMod	472	Print current module	P&rint
14	VBE	Code Window		1	Pro&ject	4		1	CodePrintProj	2557	Print all modules in proj	P&rint
15	VBE	Standard	32806	1	&Standard Form	2		1	FormNewUserf	581	Insert standardized UserForm	

Рис. 14.8. Запись в таблице меню для создания диалогового окна

После добавления строк в таблицу меню на панели Стандартная (Standard) появится следующий элемент управления (рис. 14.9).

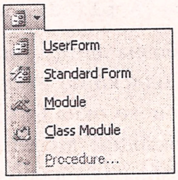


Рис. 14.9. Модифицированная панель инструментов Стандартная (Standard)

Добавьте новый модуль MenuForm и скопируйте в него следующий код:

```
Option Explicit
Option Compare Text

Private Declare Function LockWindowUpdate Lib "user32" _
  (ByVal hwndLock As Long) As Long
```

Значение свойства Application.ScreenUpdating не влияет на редактор VBE. В результате работы FormNewUserform можно наблюдать изменение размера диалогового окна и добавление элементов управления. Для фиксации окна VBE в начале процедуры и освобождения в конце можно воспользоваться простым вызовом Windows API. Дополнительная информация об использовании этой и других функций Windows API приводится в главе 16:

```
Public Sub FormNewUserForm()
    Dim component As VBIDE.VBComponent
    Dim DesignForm As UserForm

    Dim Line As Long
    Dim Button As CommandBarButton
    Dim Popup As CommandBarPopup

    Const Gap As Double = 6
```

В рекомендациях по проектированию интерфейса компания Microsoft советует размещать кнопки не ближе, чем 6 пунктов (1 пункт – 1/72 дюйма) друг от друга и от края диалогового окна.

```
On Error GoTo Catch
```

Это самая сложная подпрограмма в пределах всей надстройки, поэтому в нее будет добавлен код для обработки ошибок. Похожий код обработки ошибок должен присутствовать в каждой подпрограмме:

```
LockWindowUpdate Application.VBE.MainWindow.Hwnd
```

Для фиксации окна редактора VBE используется вызов Windows API. Обратите внимание, что `Hwnd` является скрытым свойством объекта `MainWindow`. Для отображения скрытых свойств объекта откройте окно Object Browser (Просмотр объектов), щелкните на окне правой кнопкой мыши и щелкните на пункте Show Hidden Members (Отображать скрытые компоненты).

```
Set component = Application.VBE.ActiveVBProject.VBComponents.Add( _
    vbext_ct_MSForm)
component.Properties("Width") = Application.UsableWidth * 2 / 3
component.Properties("Height") = Application.UsableHeight * 2 / 3
```

Объект `VBComponent` (в коде обозначается как `component`) предоставляет основу (фон) диалогового окна `UserForm`, коллекцию `Properties` и объект `CodeModule`. При добавлении в проект нового диалогового окна `UserForm` в качестве результата возвращается объект `VBComponent`, содержащий диалоговое окно. Коллекция `Properties` объекта `VBComponent` может использоваться для изменения размера окна (рис. 14.10) и других свойств, например цвета, шрифта и надписей на фоне.

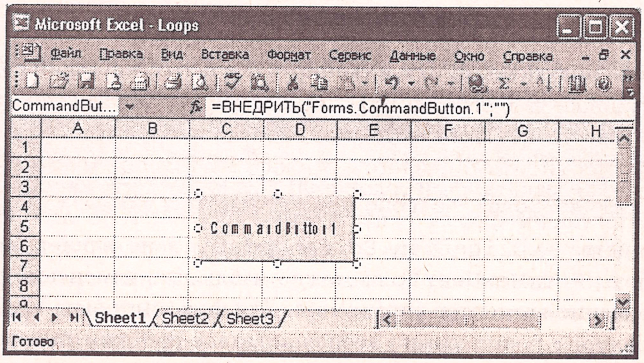


Рис. 14.10. Изменение размеров диалогового окна

```
Set DesignForm = component.Designer

    With DesignForm
        With .Controls.Add("Forms.CommandButton.1", "OKButton")
            .Caption = "OK"
            .Default = True
            .Height = 18
            .Width = 54
    End With

    With .Controls.Add("Forms.CommandButton.1", "CancelButton")
        .Caption = "Cancel"
        .Cancel = True
        .Height = 18
        .Width = 54
    End With
```

Объект `Designer` объекта `VBComponent` предоставляет доступ к содержимому диалогового окна `UserForm` и отвечает за область внутри границ диалогового окна и под строкой заголовка. Следующий код добавляет два элемента управления в пустое диалоговое окно `UserForm`. Это кнопки **OK** и **Отмена** (Cancel). Для определения имени элемента управления (в данном случае это `Forms.CommandButton.1`) можно добавить элемент управления из окна **Элементы управления** (Control Toolbox) и просмотреть результат функции `=EMBED`.

```
With .Controls("OKButton")
    .Top = DesignForm.InsideHeight - .Height - Gap
    .Left = DesignForm.InsideWidth - .Width 2 - Gap * 2
End With

With .Controls("CancelButton")
    .Top = DesignForm.InsideHeight - .Height - Gap
    .Left = DesignerForm.InsideWidth - .Width - Gap
End With
End With
```

Данную методику можно расширить на добавление списков, меток, флажков и других элементов управления. С этого момента можно работать с существующим диалоговым окном `UserForm`, менять его размер и положение, а также размер и положение элементов управления для максимального использования доступного разрешения экрана. Предыдущий код просто перемещает кнопки **OK** и **Отмена** (Cancel) в нижний правый угол диалогового окна без изменения его размера. Этот прием используется для перемещения и изменения размера всех элементов управления `UserForm`.

Теперь, когда кнопки диалогового окна `UserForm` расположены в интересующих позициях, в модуль кода диалогового окна `UserForm` можно добавить код, который будет обрабатывать события кнопок и диалогового окна. В этом примере код добавляется из строковых констант. В другой ситуации код может храниться в отдельном текстовом файле и импортироваться в модуль `UserForm`:

```
.AddFromString "Sub OKButton_Click()"
```

В данном случае добавляется код для обработки события `Click` для кнопок **OK** и **Отмена** (Cancel). Для создания подпрограммы может использоваться следующий код:

```
With component.CodeModule
    Line = .CreateEventProc("Click", "OKButton")
    .InsertLines Line, "'Стандартный обработчик кнопки OK"
    .ReplaceLine Line + 2, " mbOK = True" & vbCrLf & " Me.Hide"
```

```
    .AddFromString vbCrLf & _
    "'Стандартный обработчик кнопки Отмена" & vbCrLf & _
    "Private Sub CancelButton_Click()" & vbCrLf & _
    " mbOK = False" & vbCrLf & _
    " Me.Hide" & vbCrLf & _
    "End Sub"
```

Но если используется подпрограмма CreateEventProc, все параметры процедуры заполняются автоматически. Оба способа отличаются незначительно. Обратите внимание, что процедура CreateEventProc добавляет строку 'Private Sub...', строку 'End Sub' и пробел между ними. Не вводите этот фрагмент кода в надстройку.

```
Private Sub OKButton_Click()

End Sub
```

Подпрограмма CreateEventProc возвращает количество строк модуля, в который была вставлена строка 'Private Sub'. Это количество используется для вставки строки комментария и замены пустой строки кодом:

```
    Line = .CreateEventProc("QueryClose", "UserForm")
    .InsertLines Line, "'Стандартный обработчик события Close,"
    .InsertLines Line, "'воспринимать как обработчик кнопки Отмена"
    .ReplaceLine Line + 2, " CancelButton_Click"
    .CodePane.Window.Close
    End With
        LockWindowUpdate 0&
Exit Sub

Catch:
```

Код обработки события QueryClose для диалогового окна UserForm выглядит как обработчик кнопки **Отмена** (Cancel), поэтому код обработчика просто вызывает подпрограмму CancelButton_Click:

```
Catch:
  LockWindowUpdate 0&
  Application.Visible = False
  MsgBox Err.Description, vbExclamation
  Application.Visible = True
End Sub
```

Стандартный обработчик ошибок отключает блокировку окна, выводит сообщение об ошибке и закрывает окно. Такая обработка ошибок должна присутствовать во всех подпрограммах надстройки.

На этом создание надстройки можно считать завершенным. Переключитесь обратно в Excel и сохраните книгу как надстройку (надстройка находится в конце списка доступных типов файлов) с расширением .xla. После этого воспользуйтесь командой Сервис⇨Надстройки (Tools⇨Addins) для установки надстройки.

Работа со ссылками

Одним из значительных улучшений в последней версии VBA является возможность объявления ссылки на внешнюю объектную библиотеку (с использованием диалогового окна Tools⇨References (Сервис⇨Ссылки)). После этого определенные в библиотеке

объекты можно применять как встроенные объекты Excel. Например, в этой главе используются объекты библиотеки VBA Extensibility.

Для такого подхода существует термин "раннее связывание". Это значит, что внешняя объектная библиотека связывается с приложением на этапе проектирования. Раннее связывание предоставляет ряд преимуществ:

- код работает значительно быстрее, так как связи между библиотеками проверены и откомпилированы;
- оператор New может использоваться для создания ссылок на внешние объекты;
- можно применять все константы, определенные в объектной библиотеке, а значит, есть возможность не использовать "магические числа" в собственном коде;
- при разработке приложения Excel выводит подсказки Auto List Members, Auto Quick Info и Auto Data Tips в процессе разработки приложения.

Более подробно эта тема рассматривается в главе 16.

Но у раннего связывания есть небольшой недостаток. При попытке запуска приложения на компьютере, где не установлена объектная библиотека, на этапе компиляции будет выдана ошибка, которую невозможно перехватить стандартными методами обработки ошибок — обычно в результате перехвата в качестве ошибочной идентифицируется вполне нормальная строка. Excel выдаст сообщение об ошибке при запуске кода из модуля, который содержит:

- не определенную переменную (если в начале модуля не указан оператор `Option Explicit`);
- объявление типа определено в отсутствующей объектной библиотеке;
- константа определена в отсутствующей объектной библиотеке;
- вызов подпрограммы, объекта, метода или свойства, определенных в отсутствующей объектной библиотеке.

Коллекция `References` объекта VBE предоставляет метод проверки правильности ссылок приложения, а также проверки наличия всех необходимых объектных библиотек и правильности версий библиотек. Проверяющий код должен быть добавлен в подпрограмму `Auto_Open`, а содержащий подпрограмму `Auto_Open` модуль не должен содержать код, в котором используются внешние объектные библиотеки. Если в приложении присутствует некорректная ссылка, код вообще не будет работать и подпрограмма остановит выполнение после отображения списка некорректных ссылок. Вот типичный код подпрограммы `Auto_Open`:

```
Public Sub Auto_Open()
    Dim o As Object
    Dim IsBroken As Boolean
    Dim Description As String

    For Each o In ThisWorkbook.VBProject.References
        If o.IsBroken Then
            On Error Resume Next
            Description = "<Not known>"
            Description = o.Description
            On Error GoTo 0

            MsgBox "Отсутствующая ссылка на:" & vbCrLf & _
```

```
            "     Имя: " & Description & vbCrLf & _
            "     Путь: " & o.FullPath & vbCrLf & _
            "Пожалуйста переустановите файл."

            IsBroken = True
        End If
    Next

    If Not IsBroken Then
        ' Настройка меню
    End If
End Sub
```

Резюме

Объектная библиотека Microsoft Visual Basic for Applications Extensibility 5.3 предоставляет богатый набор объектов, свойств, методов и событий для управления самим редактором VBE. Использование этих объектов позволяет разработчикам создавать надстройки для упрощения собственной деятельности.

Многие приложения конечных пользователей также могут применять эти объекты для управления модулями кода, диалоговыми окнами `UserForm` и ссылками, что способствует повышению гибкости и эффективности.

15

Взаимодействие с другими приложениями Office

Все приложения Office (Excel, Word, PowerPoint, Outlook, Access, Publisher, SourceSafe и FrontPage) используют один и тот же язык VBA. Поняв синтаксис этого языка в Excel, его можно применять во всех других приложениях. Различия между приложениями заключаются в объектной модели.

Одной из приятных особенностей языка VBA является возможность предоставления объектов одного приложения другим приложениям. При этом взаимодействие приложений можно программировать из любого приложения. Например, для работы с объектами Word из Excel необходимо установить подключение к Word. После этого можно использовать объекты текстового процессора, как при программировании на VBA в самом приложении Word.

В этой главе рассматриваются различные способы создания связей и показаны несколько примеров программирования других приложений Microsoft. Во всех случаях код написан на языке Excel VBA, но код может быть модифицирован для использования в любом из приложений Office. Кроме того, такой код можно применять для приложений, поддерживающих язык VBA за пределами Office. Это могут быть другие продукты компании Microsoft, например Visual Basic и SQL Server. Кроме этого, растет список продуктов сторонних разработчиков, поддерживающих подобное программирование.

В конце этой главы будут рассмотрены макровирусы.

В этой главе нет подробного описания объектов, методов и свойств других приложений Office, для которых написаны примеры. В данном случае интерес вызывает только установка подключения к приложениям, а не изучение их объектной модели. Объектные модели этих приложений рассматриваются в других книгах издательства Wrox из серии Office 2000, например: Word 2000 VBA Programmer's Reference *Данкана Маккен-*

зи (Duncan MacKenzie) (ISBN: 1-861002-55-6) и Outlook 2000 VBA Programmer's Reference *Дуэйна Гиффорда (Dwayne Gifford) (ISBN: 1-861002-53-X). Кроме этого, издательство Wrox Press опубликовало подробное руководство для начинающих по программированию на Access VBA, вместе с которым поставляется компакт-диск:* Beginning Access 2000 VBA *Роба Смита (Rob Smith) и Дейва Сассмена (Dave Sussman) (ISBN: 0-7645-4383-0).*

Установка подключения

После установки подключения к приложению Office объекты приложения предоставляются для автоматизации через библиотеку типов. Существует два способа установки такого подключения: *позднее связывание (late binding)* и *ранее связывание (early binding).* В любом случае подключение устанавливается через создание объектной переменной, ссылающейся на интересующее приложение или конкретный объект в пределах интересующего приложения. После этого свойства и методы интересующего объекта можно использовать через объектную переменную.

При позднем связывании объект, ссылающийся на приложение Office, создается перед созданием ссылки на библиотеку типов приложения Office. В ранних версиях приложений Office позднее связывание было обязательным и сейчас оно продолжает использоваться, так как обладает определенными преимуществами по сравнению с ранним связыванием. Одним из преимуществ является возможность создания кода, определяющего наличие или отсутствие необходимой библиотеки типов на компьютере, а также создающего связи с разными версиями приложений в зависимости от принятых решений в процессе выполнения кода.

Недостаток позднего связывания — недоступность библиотеки типов интересующего приложения в процессе написания кода. Таким образом, отсутствует справочная информация о приложении, недоступны встроенные константы приложения, а ссылки на интересующее приложение не всегда действительны в процессе компиляции, так как их правильность невозможно проверить. Связи полностью проверяются только в процессе выполнения кода, а это требует определенных затрат времени, при этом на этапе проверки могут возникнуть ошибки, которые приведут к аварийному завершению работы приложения.

Раннее связывание поддерживается всеми версиями приложений Office начиная с Office 97. Использующий раннее связывание код работает быстрее, чем код на основе позднего связывания, так как библиотека типов интересующего приложения доступна в процессе написания кода. Таким образом, проверка синтаксиса и типов, а также связывание с библиотекой выполняются еще до запуска кода.

При использовании раннего связывания код писать намного проще, так как в окне Object Browser (Просмотр объектов) видны объекты, методы и свойства интересующего приложения, а в окне кода будут отображаться автоматические подсказки, например список свойств и методов после ввода ссылки на объект. Кроме этого, при раннем связывании можно пользоваться встроенными константами интересующего приложения.

Позднее связывание

Следующий код создает запись в календаре Outlook. В данном случае используется позднее связывание:

```
Public Sub MakeOutlookAppointment()
    Dim Outlook As Object
    Dim Appointment As Object
    Const Item = 1

    Set Outlook = CreateObject("Outlook.Application")
    Set Appointment = Outlook.CreateItem(Item)

    Appointment.Subject = "Празднование Нового года"
    Appointment.Start = DateSerial(2003, 12, 31) +
↳ TimeSerial(9, 30, 0)

    Appointment.End = DateSerial(2003, 12, 31) +
↳ TimeSerial(11, 30, 0)
    Appointment.ReminderPlaySound = True
    Appointment.Save
    Outlook.Quit
    Set Outlook = Nothing
End Sub
```

В основе метода программного управления другим приложением лежит создание объектной переменной, которая ссылается на интересующее приложение. В этом случае для удобства объектная переменная называется `Outlook`. После этого переменная `Outlook` используется для обращения к объектам в объектной модели внешнего приложения (как объект `Application` в Excel). В данном случае метод `CreateItem` объекта Outlook `Application` используется для создания объекта `AppointmentItem`.

Так как встроенные константы Outlook недоступны при позднем связывании, придется определить собственные константы, например `AppointmentItem`, или передавать в качестве параметров их значения. Обратите внимание, что время было определено через функции `DateSerial` и `TimeSerial`, что позволяет избежать проблем, связанных с интернационализацией приложений. Дополнительная информация приводится в главе 17.

Объявлением объектов `Outlook` и `Appointment` объектами универсального типа `Object` разработчик заставляет интерпретатор VBA использовать позднее связывание. Интерпретатор VBA не может построить все связи с Outlook, пока не завершится выполнение функции `CreateObject`.

Аргумент функции `CreateObject` определяет имя приложения и класс создаваемого объекта. Outlook — это имя приложения, а `Application` — класс. Многие приложения дают возможность создавать объекты на разных уровнях объектной модели. Например, Excel позволяет создавать объекты `Worksheet` и `Chart` из других приложений. Для этого в качестве аргумента функции `CreateObject` необходимо указать `Excel.Worksheet` или `Excel.Chart`.

Желательно закрывать внешнее приложение после завершения интересующих операций и присваивать объектной переменной значение `Nothing`. При этом освобождается память, выделенная для связи и приложения.

Если запустить этот макрос, в Excel вообще ничего не произойдет. Но если запустить Outlook, то в календаре можно обнаружить добавленную запись о встрече 31 декабря (рис. 5.1).

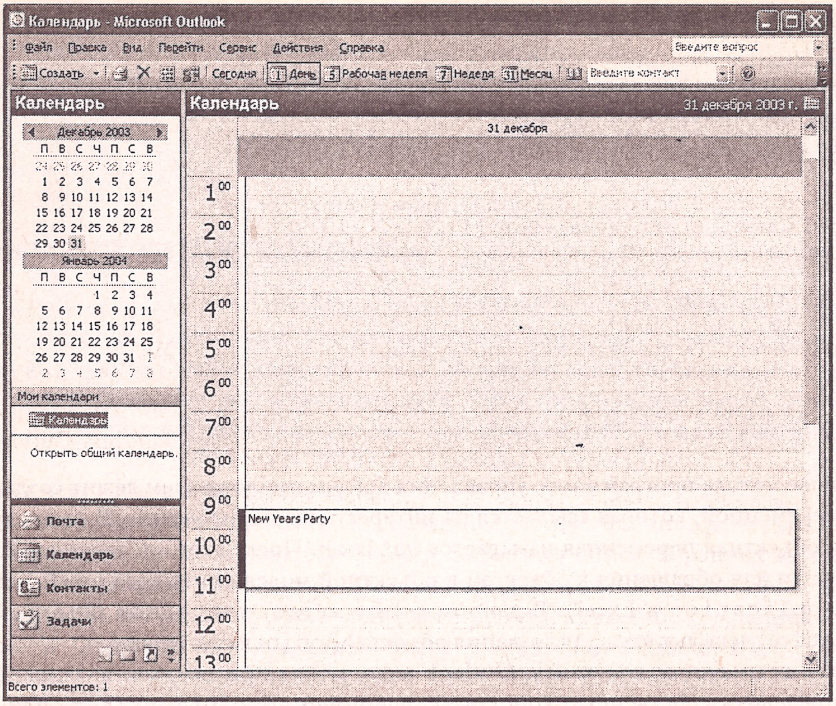


Рис. 15.1. Новая запись в календаре

Раннее связывание

Если необходимо использовать раннее связывание, в проекте VBA потребуется создать ссылку на библиотеку типов внешнего приложения. Для этого в редакторе VBE выберите пункт Tools⇨References (Сервис⇨Ссылки). Откроется диалоговое окно, показанное на рис. 15.2.

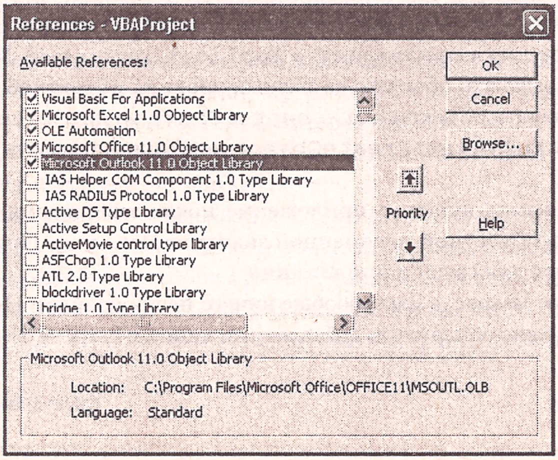


Рис. 15.2. Создание ссылок на объектные библиотеки

Для создания ссылки нужно установить флажок напротив названия объектной библиотеки. После этого можно объявлять объектные переменные интересующего типа. Например, можно объявить переменную Entry типа AddressEntry:

```
Dim Entry As AddressEntry
```

Интерпретатор VBA просмотрит библиотеки типов (сверху вниз по списку ссылок) и найдет ссылки на типы объектов. Если один и тот же тип объекта доступен в нескольких библиотеках, будет использоваться первый найденный объект. Для изменения порядка просмотра объектов можно выделить библиотеку и щелкнуть на кнопках приоритета для изменения порядка просмотра. Но полагаться на приоритет не обязательно. Объект можно квалифицировать через имя основного объекта библиотеки. Например, вместо использования AddressEntry можно указать Outlook.AddressEntry.

В следующем примере применяется раннее связывание. Код перечисляет имена записей в папке контактов Outlook. Имена записываются в столбец А активного листа. Удостоверьтесь, что перед запуском этого кода создана ссылка на объектную библиотеку Outlook:

```
Public Sub DisplayOutlookContactNames()
    Dim Outlook As Outlook.Application
    Dim NameSpace As Outlook.NameSpace
    Dim AddressList As AddressList
    Dim Entry As AddressEntry
    Dim I As Long

On Error GoTo Finally
    Set Outlook = New Outlook.Application
    Set NameSpace = Outlook.GetNamespace("MAPI")
    Set AddressList = NameSpace.AddressLists("Contacts")
    For Each Entry In AddressList.AddressEntries
        I = I + 1
        Cells(I, 1).Value = Entry.Name
    Next

Finally:
    Outlook.Quit
    Set Outlook = Nothing
End Sub
```

При попытке программного доступа к адресам электронной почты Outlook выдает предупреждение. Пользователь может разрешить или отменить операцию.

> Так как в Office XP используется улучшенная антивирусная защита, любая попытка программного доступа к адресам электронной почты приведет к выводу сообщения с предупреждением. При каждой попытке программной отправки сообщения электронной почты выдается еще одно предупреждение. Если необходимо отключить такие предупреждения, обратитесь к системному администратору.

В данном случае объектная переменная Outlook при объявлении получает тип Outlook.Application. Другие операторы Dim объявляют объектные переменные других необходимых типов. Если одно и то же имя объекта присутствует в нескольких объектных библиотеках, перед ним можно указать имя приложения и не зависеть от приоритета библиотек типов. Такой подход продемонстрирован на примере объекта Outlook.NameSpace. Для создания нового экземпляра приложения Outlook в строке присваивания ссылки Outlook.Application переменной Outlook используется ключевое слово New.

Объявление переменных определенного типа заставляет интерпретатор VBA использовать раннее связывание. Для создания объектной переменной Outlook можно воспользоваться функцией CreateObject, а не ключевым словом New, и это никак не повлияет на раннее связывание, но ключевое слово New работает более эффективно.

Открытие документа в Word

Если необходимо открыть файл, созданный в другом приложении Office, можно воспользоваться функцией GetObject для непосредственного открытия файла. С другой стороны, можно создать экземпляр приложения и открыть файл в этом приложении. Другое применения функции GetObject будет рассмотрено немного ниже.

> Для начального знакомства с объектной моделью Word можно воспользоваться механизмом записи макросов и получить список объектов, свойств и методов, которые потребуются для программного выполнения поставленной перед Word задачи.

Следующий код копирует диапазон из Excel в буфер обмена. После этого создается новый экземпляр приложения Word, открывается существующий документ Word и диапазон копируется в конец документа. Так как в коде используется ранняя привязка, удостоверьтесь, что ссылка на объектную библиотеку Word создана.

```
Public Sub CopyChartToWordDocumentCopyTableToWordDocument()
    Dim Word As Word.Application

    ThisWorkbook.Sheets("Table").Range("A1:B6").Copy

    Set Word = New Word.Application

On Error GoTo Finally

    Word.Documents.Open Filename:="C:\temp\chart.doc"
    Word.Selection.EndKey Unit:=wdStory
    Word.Selection.TypeParagraph
    Call Word.Selection.PasteSpecial(Link:=False, _
         DataType:=wdPasteOLEObject, _
        Placement:=wdInLine, DisplayAsIcon:=False)

    Call Word.ActiveDocument.Save

Finally:
    Word.Quit
    Set Word = Nothing
End Sub
```

(Удостоверьтесь, что вместо пути c:\temp\chart.doc указано реальное расположение файла документа Word.) Ключевое слово New создает новый экземпляр приложения Word, даже если приложение уже открыто. Метод Open коллекции Documents позволяет открыть существующий файл. После этого код переходит в конец документа, добавляет пустой абзац и вставляет диапазон из буфера обмена. В итоге документ сохраняется и новый экземпляр Word закрывается.

Доступ к активному документу Word

Предположим, что в Excel создана новая таблица. При этом запущено приложение Word и в нем открыт документ, в который необходимо вставить созданную таблицу. Следующий код позволяет скопировать таблицу из Excel в активный документ Word. В этом примере используется ранняя привязка:

```
Public Sub CopyTableChartToOpenWordDocument()
    Dim Word As Word.Application

    ThisWorkbook.Sheets("Table").Range("A1:B6").Copy

    Set Word = GetObject(, "Word.Application")
    Word.Selection.EndKey Unit:=wdStory

    Word.Selection.TypeParagraph
    Word.Selection.Paste

    Set Word = Nothing
End Sub
```

Функция `GetObject` принимает два необязательных аргумента. В качестве значения первого аргумента передается имя открываемого файла. В качестве второго аргумента предоставляется имя запускаемого приложения. Если не указать первый аргумент, функция `GetObject` предположит, что необходимо получить доступ к существующему экземпляру приложения. Если в качестве значения первого аргумента указать строку нулевой длины, функция `GetObject` предположит, что необходимо создать новый экземпляр Word.

В предыдущем примере показано, как использовать функцию `GetObject` без первого аргумента для обращения к текущему экземпляру Word, который расположен в памяти. Но если в памяти нет ни одного экземпляра Word, вызов функции `GetObject` без первого параметра приводит к появлению ошибки времени выполнения.

Создание нового документа Word

Предположим, что необходимо воспользоваться существующим экземпляром Word или, если его не существует, создать новый экземпляр. В любом случае необходимо открыть новый документ и вставить в него таблицу. Эту задачу выполняет следующий код. И в этом случае используется ранняя привязка:

```
Public Sub CopyTableToAnyWordDocument()
    Dim Word As Word.Application

    ThisWorkbook.Sheets("Table").Range("A1:B6").Copy
    On Error Resume Next
        ' Попытаться открыть существующий экземпляр Word
        Set Word = GetObject(, "Word.Application")

        ' Открыть новый экземпляр, если существующих
        ' экземпляров не найдено
        If Word Is Nothing Then
            Set Word = GetObject("", "Word.Application")
End If

    On Error GoTo 0
    Word.Documents.Add
```

```
    Word.Visible = True
    Word.Selection.EndKey Unit:=wdStory
    Word.Selection.TypeParagraph
    Word.Selection.Paste

    Set Word = Nothing
End Sub
```

Если в памяти отсутствуют экземпляры Word, применение функции `GetObject` без первого аргумента приводит к ошибке времени выполнения. После этого код вызывает функцию `GetObject` с нулевой строкой в качестве первого аргумента. При этом будет создан новый экземпляр Word, в котором будет открыт новый документ. Кроме этого, код делает видимым новый экземпляр Word, в то время как в предыдущих примерах все операции выполнялись в фоновом режиме без отображения диалогового окна Word. В конце процедуры объектная переменная Word освобождается, но окно Word остается на экране и позволяет просматривать результат работы кода.

Взаимодействие с Access через библиотеку DAO

Если необходимо скопировать данные из Access в Excel, можно создать ссылку на объектную библиотеку Access и воспользоваться объектной моделью Access. Кроме этого, можно воспользоваться библиотекой ADO (ActiveX Data Object), которая является наиболее актуальным интерфейсом от компании Microsoft для доступа к реляционным базам данных и другим источникам данных. Примеры использования этих интерфейсов рассматривались в главе 11.

Существует еще один метод эффективного доступа к данным в базах данных Access — библиотека DAO (Data Access Objects). В пакете Office 97 доступна только библиотека DAO, так как библиотека ADO была выпущена после Office 97. Библиотеку ADO можно использовать и в Excel 97, но мощный метод `CopyFromRecordset`, который применяется в следующем примере, в Excel 97 для наборов записей ADO не поддерживается. Ниже показано, как использовать библиотеку DAO.

На рис. 15.3 показана таблица Access, которая называется `Sales` и хранится в файле `FruitSales.mdb`.

Sales : таблица

ID	Дата	Клиент	Штат	Продукт	Количество	Цена	Доход
1	02.01.2004	Голиб	NJ	Апельсины	20	12	240
2	02.01.2004	Голиб	NJ	Манго	30	15	450
3	02.01.2004	Голиб	NJ	Манго	30	15	450
4	02.01.2004	МакДональд	MI	Бананы	50	17	850
5	02.01.2004	Киммел	MI	Яблоки	100	5	500
6	02.01.2004	Бурбонази	FL	Агава	500	10	5000
7	02.01.2004	Голиб	NJ	Грейпфруты	30	15	450
8	02.01.2004	Голиб	NJ	Оливки	30	15	450
(Счетчик)					0	0	0

Запись: 1 из 8

Рис. 15.3. Данные из таблицы Access

Следующий код использует библиотеку DAO для создания набора записей на основе таблицы `Sales`. В данном случае применяется раннее связывание, поэтому необходимо создать ссылку на объектную библиотеку DAO:

```
Public Sub GetSalesDataViaDAO()
    Dim DAO As DAO.DBEngine
    Dim Sales As DAO.Database
    Dim SalesRecordset As DAO.Recordset
    Dim I As Integer
    Dim Worksheet As Worksheet
    Dim Count As Integer

    Set DAO = New DAO.DBEngine
        Set Sales = DAO.OpenDatabase(ThisWorkbook.Path + " _
FruitSales.mdb")
    Set SalesRecordset = Sales.OpenRecordset("Sales")
    Set Worksheet = Worksheets.Add
    Count = SalesRecordset.Fields.Count
    For I = 0 To Count - 1
        Worksheet.Cells(1, I + 1).Value =
↳ SalesRecordset.Fields(I).Name
    Next

    Worksheet.Range("A2").CopyFromRecordset SalesRecordset

    Worksheet.Columns("B").NumberFormat = "mmm dd, yyyy"
    With Worksheet.Range("A1").Resize(1, Count)
        .Font.Bold = True
        .EntireColumn.AutoFit
    End With

    Set SalesRecordset = Nothing
    Set Sales = Nothing
    Set DAO = Nothing
End Sub
```

Код открывает файл базы данных Access, создает набор записей на основе таблицы `Sales` и присваивает ссылку на набор записей переменной `SalesRecordset`. В книгу Excel добавляется новый лист, а имена полей из набора записей `SalesRecordset` заносятся в первую строку нового листа. Для копирования записей из набора `SalesRecordset` в диапазон листа начиная с ячейки A2 код использует метод `CopyFromRecordset` объекта `Range`. Метод `CopyFromRecordset` предоставляет быстрый способ копирования данных (по сравнению с циклической процедурой, которая копирует каждую запись по отдельности).

Взаимодействие Access, Excel и Outlook

В качестве последнего примера интеграции приложений Office показано извлечение данных из Access, построение диаграммы в Excel и отправка диаграммы по электронной почте средствами Outlook. Код организован в четыре процедуры. Первая процедура является подпрограммой и называется `EmailChart`. Она настраивает рабочие параметры и вызывает остальные три процедуры. Обратите внимание, что в этом коде используется раннее связывание и необходимо создать ссылки на объектные библиотеки DAO и Outlook:

```
Public Sub EmailChart()
    Dim SQL As String
    Dim Range As Excel.Range
    Dim FileName As String
    Dim Recipient As String

    SQL = "SELECT Product, Sum(Revenue)"
```

```
    SQL = SQL & " FROM Sales"
    SQL = SQL & " WHERE Date>=#1/1/2004# and Date<#1/1/2005#"
    SQL = SQL & " GROUP BY Product;"

    FileName = ThisWorkbook.Path + "\Chart.xls"
    Recipient = "helge@leschinsky.in.ua"

    Set Range = GetSalesData(SQL)
    ChartData Range, FileName

    Call SendEmail(Recipient, FileName)
End Sub
```

Для хранения строки с запросом SQL используется переменная SQL. Язык запросов SQL более подробно рассматривался в главе 11. В данном случае в запросе SQL сказано, что из таблицы Sales необходимо извлечь уникальные имена продуктов и сумму прибыли для каждого продукта для всех дат в пределах 2000 года. В переменной FileName указывается путь и имя файла, которые будут использоваться для хранения книги с диаграммой. В переменной Recipient хранится адрес электронной почты, на который необходимо отправить диаграмму.

После этого код вызывает функцию GetSalesData. В качестве параметра в функцию передается запрос SQL. В результате работы функции возвращается ссылка на диапазон, содержащий извлеченные данные. Ссылка присваивается переменной Range. После этого вызывается подпрограмма ChartData. Диапазон с данными передается в подпрограмму в качестве параметра. Также в подпрограмму передаются путь и имя файла с книгой диаграммы. Наконец, вызывается подпрограмма SendEMail, в которую передается адрес электронной почты получателя и расположение книги с диаграммой, присоединяемой к сообщению.

```
Public Function GetSalesData(ByVal SQL As String) As Excel.Range
    Dim DAO As DAO.DBEngine
    Dim Sales As DAO.Database
    Dim SalesRecordset As DAO.Recordset
    Set DAO = New DAO.DBEngine

    Set Sales = DAO.OpenDatabase _
        (ThisWorkbook.Path + "\FruitSales.mdb")

    Set SalesRecordset = Sales.OpenRecordset(SQL)

    With Worksheets("Data")
        .Cells.Clear
        With .Range("A1")
            .CopyFromRecordset SalesRecordset
            Set GetSalesData = .CurrentRegion
        End With
    End With

    Set SalesRecordset = Nothing
    Set Sales = Nothing
    Set DAO = Nothing
End Function
```

Функция GetSalesData напоминает подпрограмму GetSalesDataViaDAO, которая была показана ранее. Вместо извлечения всей таблицы Sales в данном случае используется более избирательный запрос SQL. Подпрограмма очищает лист Data и копирует выделенные данные начиная с ячейки A1. Имена полей на лист не копируются. Копиру-

ются только имена продуктов и общая прибыль. Свойство CurrentRegion используется для получения ссылки на все извлеченные данные. Ссылка возвращается в качестве значения функции.

```
Public Sub ChartData(ByVal Range As Range, ByVal FileName As String)
    With Workbooks.Add
        With .Charts.Add
            With .SeriesCollection.NewSeries
                .XValues = Range.Columns(1).Value
                .Values = Range.Columns(2).Value
            End With
            .HasLegend = False
            .HasTitle = True
            .ChartTitle.Text = "Доход за 2000"
        End With
        Application.DisplayAlerts = False
        .SaveAs FileName
        Application.DisplayAlerts = True
        .Close
    End With
End Sub
```

Аргументы подпрограммы ChartData определяют диапазон с исходными данными и каталог для создаваемого файла. Подпрограмма создает новую книгу и добавляет в нее лист диаграммы. В диаграмме создается новая последовательность значений и значения из диапазона данных присваиваются осям графика. Свойство DisplayAlerts устанавливается в значение False для предотвращения предупреждений, если перезаписать существующий файл с тем же именем.

Когда вирус не является вирусом?

Авторы этой книги вспоминают, что созданная ими без злого умысла версия подпрограммы SendEmail при проверке средствами Norton AntiVirus была идентифицирована как вирус. На самом деле, код был идентифицирован как вирус X97.OutlookWorm.Gen, что отняло у них несколько часов рабочего времени. После закрытия Excel и сохранения проделанной работы Norton AntiVirus сообщил, что в книге обнаружен вирус и проблема исправлена. Антивирусный пакет удалил все модули книги. При этом пришлось отключить возможность AutoProtect в антивирусе и начинать работу с самого начала.

Вот код, который привел к появлению проблемы:

```
Public Sub SendEmail(ByVal Recipient As String,
↳ ByVal Attachment As String)
    Dim Outlook As Outlook.Application
    Dim NameSpace As Outlook.NameSpace
    Dim MailItem As Outlook.MailItem

    Set Outlook = New Outlook.Application
    Set MailItem = Outlook.CreateItem(olMailItem)
    With MailItem
        .Subject = "Диаграмма дохода за 2004"
        .Recipients.Add Recipient
        .Body = "Книга с присоединенной диаграммой"

        '.Attachments.Add Attachment
        .Send
    End With
```

```
    Set MailItem = Nothing
    Set Outlook = Nothing
End Sub
```

После ряда проб и ошибок был создан следующий код, который не идентифицируется как вирус. Код избегает идентификации как со ссылкой на объектную библиотеку Outlook, так и без нее. Так как в коде используется позднее связывание, ссылка на объектную библиотеку Outlook не требуется.

```
Public Sub SendEmail2(ByVal Recipient As String,
↳ ByVal Attachment As String)
    Dim Outlook As Object
    Dim NameSpace As Object
    Dim MailItem As Object
    Set Outlook = CreateObject("Outlook.Application")
    Set MailItem = Outlook.CreateItem(0)

    With MailItem
        .Subject = "Диаграмма дохода за 2004 год"
        .Recipients.Add Recipient
        .Body = "Книга с присоединенной диаграммой"
        .Attachments.Add Attachment
        .Send
    End With
End Sub
```

Подпрограмма SendEmail принимает адрес электронной почты и имя файла в качестве параметров. Если конфигурация Outlook требует предварительной регистрации, придется удалить символ комментария в начале строки получения ссылки на NameSpace и предоставить имя и пароль. Новое сообщение электронной почты создается с помощью метода CreateItem. После этого указывается текст темы письма и присоединяемый файл. Метод Send позволяет отправить сообщение электронной почты.

При выполнении этого кода в Office XP потребуется ответить на запросы трех диалоговых окон. В первых двух выдается предупреждение о попытке доступа к Outlook, потом во втором диалоговом окне устанавливается пятисекундная задержка и выдается предупреждение об отправке сообщения электронной почты. Эта методика очень полезна как для легальной деятельности, так и для написания вирусов.

Хотя Office 2003 обеспечивает стойкую защиту против вирусов на основе электронной почты, самые ранние версии Office оказываются более уязвимы. Компания Microsoft предоставляет исправления для ранних версий Outlook, но они могут полностью отключить возможность программного создания сообщений электронной почты средствами Outlook. Очень сложно дать разрешение на программное создание сообщений электронной почты и при этом запретить вирусам делать то же самое. Лучшим решением является установка последней версии антивирусного программного обеспечения.

Резюме

Чтобы автоматизировать использование объектов другого приложения, необходимо создать объектную переменную, которая ссылается на интересующее приложение или объект интересующего приложения. Для создания связи между VBA и объектами другого приложения можно использовать раннее или позднее связывание. Раннее связывание требует создания ссылки на библиотеку типов интересующего приложения. При этом

объектные переменные, ссылающиеся на объекты приложения, должны объявляться с использованием правильного типа. Если объявить объектную переменную с использованием универсального типа `Object`, интерпретатор VBA будет применять позднее связывание.

При раннем связывании генерируется более быстрый код, чем при использовании позднего связывания. Кроме этого, информация об объектах интересующего приложения доступна в окне Object Browser (Просмотр объектов), а при наборе кода автоматически выводятся всплывающие подсказки. Кроме того, в процессе ввода кода выполняется проверка синтаксиса и типов, а значит, в процессе выполнения будет выдаваться меньше ошибок, чем при позднем связывании, когда все эти проверки не могут быть выполнены до начала работы кода.

При использовании позднего связывания для создания ссылок объектных переменных на целевое приложение применяются функции `CreateObject` и `GetObject`. Эти же функции можно использовать при раннем связывании, но ключевое слово New работает более эффективно. Однако если необходимо проверить открытый экземпляр другого приложения на этапе выполнения, функцию `GetObject` можно использовать и вместе с ранним связыванием.

Показанные в этой главе методики позволяют создавать мощные приложения, которые хорошо интегрируются в уникальные возможности различных продуктов. Пользователь остается в знакомой среде, например в Excel, а код работает с несколькими продуктами, предоставляющими свои объекты через библиотеки типов.

Стоит помнить, что создатели вирусов могут использовать показанную здесь информацию для нападения на незащищенные системы. Удостоверьтесь в адекватной защите собственных систем.

16

Программирование с помощью Windows API

Язык Visual Basic for Applications является языком высокого уровня и предоставляет богатый, но в то же время простой набор функциональности для управления пакетом Office, а также другими приложениями. При этом разработчик защищен от программирования операционной системы Windows, которым приходится заниматься разработчикам на C++.

Цена этой защиты – невозможность контроля над многими компонентами платформы Windows. Например, функция `Application.International` используется для чтения большинства региональных параметров Windows, а `Application.UsableWidth` и `Application.UsableHeight` позволяют определить размеры экрана. Вся связанная с Windows информация доступна в виде свойств объекта `Application`. Эти свойства перечислены в приложении А.

Платформа Windows предоставляет большой объем низкоуровневой функциональности, которая обычно не доступна из VBA. Эта функциональность позволяет определять системные цвета, создавать временные файлы и т.д. Ограниченное подмножество функциональности предоставляется в VBA, например, дается возможность создания и использования подключения к сети Internet. В частности, с помощью вызова `Workbooks.Open "<URL>"` можно открыть страницу в сети Internet, но сохранение страницы на диске невозможно. Кроме этого, существует ряд объектных библиотек, доступных на компьютере Windows для предоставления высокоуровневого доступа к функциональности Windows для приложений VBA. Например, существуют библиотеки Windows Scripting Runtime и Internet Transfer Control.

Но иногда возникают ситуации, в результате которых необходимо выйти за пределы VBA и объектных библиотек и погрузиться в файлы с низкоуровневыми процедурами, предоставляемыми и используемыми Windows. Операционная система Windows состоит

из большого количества отдельных файлов, большинство из которых являются библиотеками DLL и содержат ограниченный набор связанных друг с другом функций. Функции из библиотек DLL могут использоваться операционной системой Windows и другими библиотеками DLL. Непосредственный запуск библиотек DLL невозможен.

Все эти файлы известны как программный интерфейс приложений Windows (Windows API). Вот наиболее часто используемые файлы из состава Windows API:

Файл	*Группы функций*
`USER32.DLL`	Функции пользовательского интерфейса (управление окнами, клавиатурой и буфером обмена)
`KERNEL32.DLL`	Файловые и системные функции (например, для управления программами)
`GDI32.DLL`	Графические и экранные функции
`SHELL32.DLL`	Функции оболочки Windows (например, обработка пиктограмм и запуск программ)
`COMDLG32.DLL`	Стандартные функции диалоговых окон Windows
`ADVAPI32.DLL`	Функции системного реестра и подсистемы NT Security
`MPR.DLL` и `NETAPI32.DLL`	Сетевые функции
`WININET.DLL`	Функции работы с сетью Internet
`WINMM.DLL`	Функции мультимедиа
`WINSPOOL.DRV`	Функции печати

В этой главе рассматривается использование функций из этих файлов в приложениях VBA и предоставляется несколько полезных примеров. Все функции Windows API документированы в Platform SDK библиотеки MSDN Library по адресу `http://msdn.microsoft.com/library/default.asp`. Информацию по этому адресу можно рассматривать как интерактивное справочное руководство по Windows API.

Анатомия вызова программного интерфейса приложений

Перед использованием процедур, входящих в библиотеку Windows DLL, интерпретатору VBA необходимо сообщить расположение библиотеки, параметры процедур и типы возвращаемых значений. Для этого применяется оператор `Declare`, который описан в справочном руководстве VBA следующим образом:

```
[Public | Private] Declare Sub имя Lib "имя_библиотеки" [Alias
"имя_псевдонима"] [([список_аргументов])]
[Public | Private] Declare Function name Lib "имя_библиотеки" [Alias
"имя_псевдонима"] [([список_аргументов])] [As тип]
```

Справочное руководство VBA содержит хорошее описание синтаксиса этих операторов, но в нем нет ни одного примера вызова метода API. Следующее объявление используется для обнаружения каталога Windows TEMP:

```
Private Declare Function GetTempPath Lib "kernel32" _
        Alias "GetTempPathA" ( _
        ByVal nBufferLength As Long, _
        ByVal lpBuffer As String) As Long
```

Предыдущее объявление сообщает интерпретатору VBA, что:

- функция вызывается из кода как GetTempPath;
- процедура API находится в библиотеке kernel32.dll;
- в данном случае используется метод GetTempPathA (имя чувствительно к регистру);
- функция GetTempPath принимает два параметра типов Long и String (дополнительная информация о типах приводится далее);
- функция GetTempPath возвращает значение типа Long.

Объявления большинства распространенных функций API доступны в файле win32api.txt, предоставляемом в составе Office XP Developer Edition и в любой более новой версии Visual Basic. Кроме этого, предоставляется небольшой аплет API Viewer, который поддерживает поиск объявлений. На момент написания книги этот текстовый файл был доступен на странице "Free Stuff" раздела Office Developer на сайте компании Microsoft по адресу http://www.microsoft.com/officedev/o-free.htm или непосредственно в разделе http://www.microsoft.com/downloads (выполните поиск по ключевому слову win32api.exe).

Интерпретация объявлений в стиле C

Библиотека MSDN является лучшим источником информации о функциях в библиотеке Windows API, но в основном эта информация ориентирована на программистов C и C++. В библиотеке предоставляются объявления функций в стиле C. В файле win32api.txt приводятся объявления большинства основных функций Windows в стиле VBA, но этот список не обновлялся некоторое время и поэтому не содержит части новых функций из библиотек Windows DLL (например, функций OLE из библиотек olepro32.dll и функций работы с сетью Internet из библиотеки WinInet.dll). Обычно можно преобразовать объявления в стиле C в объявления VBA Declare. Для этого используется показанный ниже метод.

В библиотеке MSDN показано следующее объявление функции GetTempPath (по адресу http://msdn.microsoft.com/library/default.asp?url=/library/en-us/fileio/base/gettemppath.asp):

```
DWORD GetTempPath(
    DWORD nBufferLength, // размер буфера в символах
    LPTSTR lpBuffer      //  указатель на буфер с именем каталога
);
```

Это объявление можно воспринимать следующим образом:

```
<Возвращаемый тип> <Имя функции>(
    <Тип данных параметра> <Имя параметра>,
    <Тип данных параметра> <Имя параметра>,
);
```

Преобразование объявления в стиле C в оператор VBA `Declare` позволяет получить следующий результат (ключевые слова DWORD и LPTSTR преобразовываются в типы данных VBA):

```
Declare Function имя_функции Lib "имя_библиотеки" Alias "GetTempPathA"
(ByVal nBufferLength As DWORD, ByVal lpBuffer As LPSTR) As DWORD
```

На платформе Windows существует два набора символов. Набор символов ANSI был стандартом в течение многих лет и использует один байт для описания одного символа, что в любой момент обеспечивает доступность 255 символов. Для предоставления одновременного доступа к большему диапазону символов (например при использовании дальневосточных алфавитов) был создан набор символов Unicode. В этом наборе для каждого символа выделяется два байта, что позволяет одновременно иметь доступ к 65535 символам.

Для предоставления одной и той же функциональности для обоих наборов символов в составе Windows API предлагается две версии каждой функции, работающей со строками. Функции для набора символов ANSI имеют суффикс `A`, а функции для набора символов Unicode — суффикс `W`. Интерпретатор VBA всегда использует строки ANSI, поэтому тут все время будут применяться функции с суффиксом `A` — в данном случае `GetTempPathA`. Кроме этого, объявления в стиле C используют другие имена для обозначения типов данных (их тоже придется преобразовать). В следующей таблице показаны наиболее распространенные типы данных (хотя этот список не полный):

Тип данных в C	*Объявление в стиле VBA*
`BOOL`	`ByVal <Имя> As Long`
`BYTE`	`ByVal <Имя> As Byte`
`BYTE *`	`ByRef <Имя> As Byte`
`Char`	`ByVal <Имя> As Byte`
`Char_huge *`	`ByVal <Имя> As String`
`Char FAR *`	`ByVal <Имя> As String`
`Char NEAR *`	`ByVal <Имя> As String`
`DWORD`	`ByVal <Имя> As Long`
`HANDLE`	`ByVal <Имя> As Long`
`HBITMAP`	`ByVal <Имя> As Long`
`HBRUSH`	`ByVal <Имя> As Long`
`HCURSOR`	`ByVal <Имя> As Long`
`HDC`	`ByVal <Имя> As Long`
`HFONT`	`ByVal <Имя> As Long`
`HICON`	`ByVal <Имя> As Long`
`HINSTANCE`	`ByVal <Имя> As Long`
`HLOCAL`	`ByVal <Имя> As Long`
`HMENU`	`ByVal <Имя> As Long`

Тип данных в C	*Объявление в стиле VBA*
HMETAFILE	ByVal <Имя> As Long
HMODULE	ByVal <Имя> As Long
HPALETTE	ByVal <Имя> As Long
HPEN	ByVal <Имя> As Long
HRGN	ByVal <Имя> As Long
HTASK	ByVal <Имя> As Long
HWND	ByVal <Имя> As Long
Int	ByVal <Имя> As Long
int FAR *	ByRef <Имя> As Long
LARGE_INTEGER	ByVal <Имя> As Currency
LONG	ByVal <Имя> As Long
LPARAM	ByVal <Имя> As Long
LPCSTR	ByVal <Имя> As String
LPCTSTR	ByVal <Имя> As String
LPSTR	ByVal <Имя> As String
LPTSTR	ByVal <Имя> As String
LPVOID	ByRef <Имя> As Any
LRESULT	ByVal <Имя> As Long
UINT	ByVal <Имя> As Integer
UINT FAR *	ByRef <Имя> As Integer
WORD	ByVal <Имя> As Integer
WPARAM	ByVal <Имя> As Integer
Другой	Возможно, определенный пользователем тип, который придется определить самостоятельно

Некоторые определения API в библиотеке MSDN содержат идентификаторы IN и OUT. Если тип VBA показан в таблице 'ByVal <Имя> As Long', то для параметров OUT он должен быть изменен на 'ByRef...'.

Обратите внимание, что в функции API строки всегда передаются по значению (ByVal). Это связано с тем, что интерпретатор VBA использует собственный механизм хранения строк, который не поддерживается библиотеками DLL для языка C. При передаче строк по значению интерпретатор VBA выполняет преобразование собственной структуры хранения в структуру, понятную библиотеке DLL.

Учитывая это, после преобразования и удаления ненужных префиксов получается следующее объявление:

```
Private Declare Function GetTempPath Lib "имя_библиотеки" _
 Alias "GetTempPathA" (ByVal nBufferLength As Long, _
ByVal Buffer As String) As Long
```

Однако в объявлении не содержится информация о библиотеке DLL, в которой хранится интересующая функция. В нижней части страницы MSDN в разделе "Требования" содержится строка:

```
Library: Use kernel32.lib.
```

Это говорит о том, что функция хранится в файле kernel32.dll. Финальное объявление будет выглядеть следующим образом:

```
Private Declare Function GetTempPath Lib "kernel32.dll" _
 Alias "GetTempPathA" (ByVal nBufferLength As Long, _
 ByVal Buffer As String) As Long
```

Полученное объявление функции практически идентично объявлению из файла win32api.txt. Этот файл стоит использовать в качестве основного справочного руководства по объявлениям всех функций API.

> Внимание: использование неправильного объявления функции может привести к аварийному завершению работы Excel. При разработке с применением вызовов API как можно чаще сохраняйте проделанную работу.

Константы, структуры, обработчики и классы

Большинство функций API имеют аргументы, принимающие ограниченное количество предопределенных значений. Например, для получения информации о возможностях операционной системы можно воспользоваться функцией GetSystemMetrics:

```
Declare Function GetSystemMetrics Lib "user32" ( _
        ByVal Index As Long) As Long
```

Обратите внимание, что в файле win32api.txt функция GetSystemMetrics показана с применением ключевого слова Alias:

```
Declare Function GetSystemMetrics Lib "user32" _
Alias "GetSystemMetrics" (ByVal nIndex As Long) As Long
```

Ключевое слово Alias не понадобится в том случае, если имя функции совпадает с псевдонимом, поэтому оно автоматически удаляется при копировании функции в модуль кода.

Значение аргумента Index сообщает функции об интересующей метрике. Аргументу Index присваивается одна из предопределенных констант. Соответствующие константы показаны в документации MSDN, но чаще всего значения констант не показываются. К счастью, в файле win32api.txt содержится большинство необходимых констант. Для функции GetSystemMetrics существует около 80 констант, включая SM_CXSCREEN и SM_CYSCREEN, которые позволяют получить размеры экрана:

```
Private Const SM_CXSCREEN As Long = 0
Private Const SM_CYSCREEN As Long = 1

Private Declare Function GetSystemMetrics Lib "user32" _
                (ByVal Index As Long) As Long

Public Sub ShowScreenDimensions()
    Dim X As Long
    Dim Y As Long
```

```
    X = GetSystemMetrics(SM_CXSCREEN)
    Y = GetSystemMetrics(SM_CYSCREEN)

    Call MsgBox("Разрешение экрана составляет " & X & "x" & Y)
End Sub
```

Многие функции Windows API передают информацию с помощью *структур (structures)*. Это термин языка C, который соответствует определенному пользователем типу (User-Defined Type — UDT). Например, функция `GetWindowRect` применяется для получения размера экрана:

```
Declare Function GetWindowRect Lib "user32" ( _
    ByVal hwnd As Long, _
    ByRef lpRect As Rect) As Long
```

Параметр `lpRect` является структурой RECT, которая заполняется функцией `GetWindowRect` и содержит информацию о размерах окна. В библиотеке MSDN показано следующее определение структуры RECT:

```
typedef struct tagRECT {
    LONG left;
    LONG top;
    LONG right;
    LONG bottom;
} RECT;
```

Это объявление может быть преобразовано в определенный пользователем тип VBA с использованием того же преобразования типов данных, которое было показано в предыдущем разделе:

```
Private Type Rect
    Left As Long
    Top As Long
    Right As Long
    Bottom As Long
End Type
```

В файле `win32api.txt` предоставляются определения пользовательских типов данных для большинства распространенных структур.

Первый параметр функции `GetWindowRect` показан как `'hwnd'` и представляет дескриптор окна. Дескриптор просто является указателем на область памяти, в которой содержится информация об интересующем объекте (в этом случае, окно). Операционная система Windows динамически выделяет дескрипторы и вероятность их повторения в различных сеансах крайне мала. Следовательно, номер дескриптора невозможно непосредственно использовать в коде и поэтому необходимо применять функции API, возвращающие необходимый дескриптор. Например, для получения размеров окна Excel необходимо получить дескриптор окна Excel. Этот дескриптор предоставляет функция API `FindWindow`.

```
' Вызов API для поиска окна
Private Declare Function FindWindow Lib "user32" _
    Alias "FindWindowA" (ByVal ClassName As String, _
    ByVal WindowName As String) As Long
```

Эта функция просматривает все открытые окна, пока не найдет окно с указанными именем класса и заголовком. В Excel 2002 свойство `hWnd` было добавлено в объект `Application`, поэтому для этих целей функцию `FindWindow` можно больше не приме-

нять. В настоящей главе во всех примерах используется функция `FindWindow`, что позволяет сохранить совместимость с предыдущими версиями Excel.

Приложения Windows поддерживают множество различных типов окон, начиная от окна приложения Excel и заканчивая окнами диалоговых листов, диалоговых окон `UserForm`, окон списков или кнопок. Каждый тип окна имеет уникальный идентификатор, называемый *классом (class)*. Вот распространенные имена классов в Excel:

Окно	Имя класса
Основное окно Excel	`XLMAIN`
Лист Excel	`EXCEL7`
Диалоговое окно UserForm	`ThunderDFrame` (в Excel 2003, Excel 2002 и Excel 2000)
	`ThunderRT6DFrame` (в Excel 2003, Excel 2002 и Excel 2000 при запуске в качестве надстройки COM Addin)
	`ThunderXFrame` (в Excel 97)
Диалоговый лист Excel	`bosa_sdm_xl9` (в Excel 2002 и Excel 2000)
	`bosa_sdm_xl8` (в Excel 97)
	`bosa_sdm_xl` (в Excel 5 и Excel 95)
Строка состояния Excel	`EXCEL4`

Функция `FindWindow` ищет окно по имени класса и тексту заголовка.

Обратите внимание, что имена классов для некоторых стандартных элементов меняются с каждой новой версией Excel. Таким образом, код должен проверять текущую версию для выбора соответствующего имени класса:

```
Select Case Val(Application.Version)
    Case Is >= 11 'Использовать имена классов Excel 2003
    Case Is >= 9  'Использовать имена классов Excel 2000/2002
    Case Is >= 8  'Использовать имена классов Excel 97
    Case Else     'Использовать имена классов Excel 5/95
End Select
```

Это приводит к проблеме совместимости сверху вниз. Было бы неплохо писать код с определенной степенью уверенности, что он будет работать и в последующих версиях Excel, но заранее неизвестно, какие имена классов будут использоваться в следующих версиях. К счастью, пока имена классов не менялись с момента выхода Excel 2000.

Учитывая все вышеизложенное, можно воспользоваться следующим кодом для определения расположения и размера основного окна Excel (в пикселях):

```
Private Declare Function FindWindow Lib "user32" _
    Alias "FindWindowA" (ByVal ClassName As String, _
    ByVal WindowName As String) As Long
Declare Function GetWindowRect Lib "user32" ( _
    ByVal hWnd As Long, _
    Rect As Rect) As Long

Private Type Rect
    Left As Long
    Top As Long
    Right As Long
```

```
    Bottom As Long
End Type

Public Sub ShowExcelWindowSize()
    Dim hWnd As Long, aRect As Rect
    hWnd = FindWindow("XLMAIN", Application.Caption)
    Call GetWindowRect(hWnd, aRect)
    Call PrintRect(aRect)
End Sub

Private Sub PrintRect(ByRef aRect As Rect)
    Call MsgBox("Окно Excel имеет следующие границы:" & _
        vbCrLf & " Левая: " & aRect.Left & _
        vbCrLf & " Правая: " & aRect.Right & _
        vbCrLf & " Верхняя: " & aRect.Top & _
        vbCrLf & " Нижняя: " & aRect.Bottom & _
        vbCrLf & " Ширина: " & (aRect.Right - aRect.Left) & _
        vbCrLf & " Высота: " & (aRect.Bottom - aRect.Top))
End Sub
```

Измените размеры окна Excel, чтобы оно занимало часть экрана, и запустите подпрограмму `ShowExcelWindowSize`. Появится диалоговое окно, в котором показаны границы и размеры окна Excel. Разверните окно Excel во весь экран и выполните процедуру еще раз — верхняя и левая границы могут оказаться отрицательными. Это связано с тем, что функция `GetWindowRect` возвращает размер окна Excel по внешним границам окна. При разворачивании окна во весь экран границы оказываются за пределами видимой области, но остаются частью окна.

Что делать, если что-то пошло не так?

Одним из недостатков использования функций Windows API является идентификация причины ошибки. Если по какой-либо причине вызов API завершился неудачно, он должен возвратить какой-то признак неудачного завершения (обычно нулевое значение функции) и зарегистрировать ошибку в операционной системе Windows. После этого средствами функции VBA `Err.LastDLLError` можно извлечь код ошибки и воспользоваться функцией `FormatMessage` из Windows API для получения текста с описанием ошибки.

```
Private Const FORMAT_MESSAGE_FROM_SYSTEM As Long = &H1000

Private Declare Function FindWindow Lib "user32" _
    Alias "FindWindowA" (ByVal ClassИмя As String, _
    ByVal WindowИмя As String) As Long
Declare Function GetWindowRect Lib "user32" ( _
    ByVal hWnd As Long, _
    Rect As Rect) As Long

Private Declare Function FormatMessage Lib "kernel32" _
    Alias "FormatMessageA" (ByVal dwFlags As Long, _
    ByVal Source As Long, ByVal MessageId As Long, _
    ByVal LanguageId As Long, ByVal Buffer As String, _
    ByVal Size As Long, ByVal Arguments As Long) As Long

Private Type Rect
    Left As Long
    Top As Long
    Right As Long
```

```
    Bottom As Long
End Type

Private Sub PrintRect(ByRef aRect As Rect)
    Call MsgBox("Границы и размеры:" & _
        vbCrLf & " Левая: " & aRect.Left & _
        vbCrLf & " Правая: " & aRect.Right & _
        vbCrLf & " Верхняя: " & aRect.Top & _
        vbCrLf & " Нижняя: " & aRect.Bottom & _
        vbCrLf & " Ширина: " & (aRect.Right - aRect.Left) & _
    vbCrLf & " Высота: " & (aRect.Bottom - aRect.Top))
End Sub

Sub ShowExcelWindowSize()
    Dim hWnd As Long
    Dim aRect As Rect

    hWnd = FindWindow("XLMAIN", Application.Caption)
    If hWnd = 0 Then
        Call MsgBox(LastDLLErrText(Err.LastDllError))
    Else
        Call GetWindowRect(hWnd, aRect)
        Call PrintRect(aRect)
    End If
End Sub

Function LastDLLErrText(ByVal ErrorCode As Long) As String
    Dim Buffer As String * 255
    Dim Result As Long

  Result = FormatMessage(FORMAT_MESSAGE_FROM_SYSTEM, 0&, ErrorCode, _
    0, Buffer, 255, 0)

  LastDLLErrText = Left(Buffer, Result)
End Function
```

Полный код этого примера находится в модуле `Module3` в книге `Examples.xls`, доступной на сайте Wrox по адресу `http://www.wrox.com`.

К сожалению, в разных версиях Windows функция `Err.LastDllError` инициализируется различными значениями. Например, если в предыдущем примере в вызове функции `FindWindow` изменить имя класса на `XLMAINTEST`, можно предположить, что в результате возникнет ошибка `"Невозможно найти окно"` ("Unable to find window"). Именно такое сообщение появится в операционной системе Windows NT 4.0. Но в операционной системе Windows 98 информация об ошибке не предоставляется и возвращается стандартное сообщение `"Операция успешно завершена"` ("The operation completed successfully"). В большинстве случаев информация об ошибке предоставляется, как будет показано в следующей главе.

Сокрытие вызовов API в модулях классов

Если в приложении приходится использовать большое количество вызовов API, код может очень быстро потерять стройную структуру. Большинство разработчиков предпочитают скрывать вызовы API в модулях классов. Такое решение предоставляет ряд преимуществ, а именно:

- объявления и вызовы API скрываются от основного кода приложения;
- модуль класса может выполнять операции по инициализации и очистке, улучшая стабильность приложения;
- большинство вызовов API принимают большое количество параметров. Некоторые из этих параметров имеют одинаковое значение при всех вызовах функции. Модуль класса может предоставлять доступ только к тем параметрам, которые должны меняться приложением;
- модули классов могут храниться в виде текстовых файлов или в Code Librarian (при использовании Office Developer). Модуль класса предоставляет выделенный набор функциональности, который может использоваться в будущих приложениях.

В следующем примере показан код модуля класса для работы с временными файлами. Предоставляется следующая функциональность:

- создание временных файлов в принятом по умолчанию временном каталоге Windows (`TEMP`);
- создание временного файла в указанном пользователем каталоге;
- получение пути и имени временного файла;
- получение текста ошибок, которые могли произойти при создании временного файла;
- удаление временного файла.

Создайте модуль класса, который называется `TempFile`. Скопируйте в модуль следующий код (этот класс также доступен в книге `Examples.xls`, которую можно загрузить на сайте издательства Wrox по адресу `http://www.wrox.com`).

```
Option Explicit

Private Declare Function GetTempPath Lib "kernel32" _
    Alias "GetTempPathA" ( _
    ByVal BufferLength As Long, _
    ByVal Buffer As String) As Long

Private Declare Function GetTempFileName Lib "kernel32" _
    Alias "GetTempFileNameA" ( _
    ByVal Path As String, _
    ByVal PrefixString As String, _
    ByVal Unique As Long, _
    ByVal TempFileName As String) As Long

Private Declare Function FormatMessage Lib "kernel32" _
    Alias "FormatMessageA" ( _
    ByVal Flags As Long, _
    ByVal Source As Long, _
    ByVal MessageId As Long, _
    ByVal LanguageId As Long, _
    ByVal Buffer As String, _
    ByVal Size As Long, _
    ByVal Arguments As Long) As Long

Const FORMAT_MESSAGE_FROM_SYSTEM As Long = &H1000

Dim TempPath As String
```

```
Dim TempFile As String
Dim ErrorMessage As String
Dim TidyUp As Boolean
```

Одним из преимуществ использования модуля класса является возможность выполнения операций на этапе инициализации. В данном случае на этапе инициализации определяется расположение каталога TEMP. Если пользователь не указал собственный каталог, временные файлы создаются в каталоге TEMP.

```
Private Sub Class_Initialize()
    Dim Buffer As String * 255
    Dim Result As Long
    Result = GetTempPath(255, Buffer)

    If Result = 0 Then
        ErrorMessage = LastDLLErrText(Err.LastDllError)
    Else
        TempPath = Left(Buffer, Result)
    End If
End Sub
```

Эта подпрограмма создает временный файл и возвращает его имя (включая путь). В самом простом случае для создания временного файла достаточно вызвать этот метод:

```
Public Function CreateFile() As String
    Dim Buffer As String * 255
    Dim Result As Long

    Result = GetTempFileName(TempPath, "", 0, Buffer)

    If Result = 0 Then
        ErrorMessage = LastDLLErrText(Err.LastDllError)
    Else
        TempFile = Left(Buffer, InStr(1, Buffer, Chr(0)) - 1)
        ErrorMessage = "OK"
        TidyUp = True
        CreateFile = TempFile
    End If
End Function
```

Модуль класса может предоставлять набор свойств, которые позволяют вызывающей подпрограмме получать и модифицировать параметры создания временных файлов. Например, вызывающей программе позволено выбрать каталог для создания временного файла. Это поведение можно расширить, сделав свойство доступным только для чтения после создания временного файла. Тогда при попытке модификации свойства будет выдаваться сообщение об ошибке. Использование процедур Property в модулях классов рассматривалось в главе 6.

```
Public Property Get Path() As String
    Path = Left(TempPath, Len(TempPath) - 1)
End Property

Public Property Let Path(ByVal NewPath As String)
    TempPath = NewPath
    If Right(TempPath, 1) <> "\" Then
        TempPath = TempPath & "\"
    End If
End Property
```

Кроме этого, вызывающей подпрограмме предоставляется возможность чтения имени и полного имени (включающего путь) временного файла.

```
Public Property Get Name() As String
   Name = Mid(TempFile, Len(TempPath) + 1)
End Property

Public Property Get FullName() As String
   FullName = TempFile
End Property
```

Также подпрограмме предоставляется возможность чтения сообщений об ошибках:

```
Public Property Get ErrorText() As String
  ErrorText = ErrorMessage
End Property
```

Вызывающей программе предоставляется возможность удаления временных файлов после завершения работы:

```
Public Sub Delete()
   On Error Resume Next
   Kill TempFile
   TidyUp = False
End Sub
```

По умолчанию созданные временные файлы удаляются при удалении экземпляра класса. При этом вызывающее приложение может потребовать другого поведения, поэтому стоит предоставить параметры, управляющие этим поведением:

```
Public Property Get TidyUpFiles() As Boolean
   TidyUpFiles = TidyUp
End Property

Public Property Let TidyUpFiles(ByVal IsNew As Boolean)
   TidyUp = IsNew
End Property
```

В коде обработки события `Terminate` выполняется удаление временных файлов, если вызывающий код не потребовал обратного. Этот код выполняется при уничтожении экземпляра класса. Если экземпляр объявлялся внутри подпрограммы, он будет уничтожен при выходе объектной переменной из области видимости в момент завершения работы подпрограммы. Если экземпляр объявлен на уровне модуля, этот код будет вызываться при закрытии книги:

```
Private Sub Class_Terminate()
   If TidyUp Then Delete
End Sub
```

Для извлечения текста ошибки Windows API используется та же функция, которая была показана в предыдущем разделе.

```
Private Function LastDLLErrText(ByVal ErrorCode As Long) As String
   Dim Buffer As String *255
   Dim Result As Long

   Result = FormatMessage(FORMAT_MESSAGE_FROM_SYSTEM, _
   0&, ErrorCode, 0, Buffer, 255, 0)

   LastDLLErrText = Left(Buffer, Result)
End Function
```

После того как модуль класса включен в проект, вызывающая процедура может ничего не знать о функциях Windows API:

```
Public Sub TestTempFile()
    Dim Object As New TempFile

    If Object.CreateFile = "" Then
        Call MsgBox("При создании временного файла произошла
↳ошибка:" & vbCrLf & obTempFile.ErrorText)
    Else
        Call MsgBox("Создан временный файл " & Object.FullName )
    End If
End Sub
```

В Windows XP (и в других версиях Windows) будет получен следующий результат:

```
Создан временный файл C:\WINDOWS\TEMP\5024.TMP
```

Обратите внимание, что временный файл создан во время работы функции `CreateFile`. После завершения процедуры переменная `Object` выходит за пределы области видимости и уничтожается интерпретатором VBA. Удаление файла обеспечивается процедурой обработки события `Terminate` — вызывающая процедура может ничего не знать о подпрограммах удаления временных файлов. Если подпрограмму `CreateFile` вызывать несколько раз, будет удален только последний временный файл. Для каждого временного файла необходимо создавать новый экземпляр класса.

Для провоцирования сообщения об ошибке в подпрограмме `TestTempFile` можно указать несуществующий каталог для создания временного файла:

```
Public Sub TestTempFile()
    Dim Object As New TempFile
    Object.Path = "C:\NoSuchPath"

    If Object.CreateFile = "" Then
        Call MsgBox("В процессе создания временного файла " & _
        Chr(10) & Object.ErrorText & Chr(10) & " произошла ошибка")
    Else
        Call MsgBox("Создан временный файл " & Object.FullName)
    End If
End Sub
```

В этот раз выдается осмысленное сообщение об ошибке (рис. 16.1).

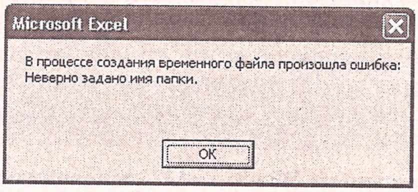


Рис. 16.1. Сообщение, которое выдается при ошибке во время создания временного файла

Примеры классов

В этом разделе рассматриваются некоторые распространенные вызовы API. Обратите внимание, что в каждом случае определения функций и констант должны указываться в разделе `Declarations` в начале модуля.

Класс таймера высокого разрешения

При тестировании кода может потребоваться измерение времени выполнения различных подпрограмм. Обычно такое измерение требуется для определения узких мест в производительности. Интерпретатор VBA предоставляет две функции, которые могут использоваться в качестве таймера:

- функция Now возвращает текущее время и имеет разрешающую способность в 1 секунду;
- функция Timer возвращает количество миллисекунд с последней полуночи и имеет разрешающую способность примерно в 10 миллисекунд.

Ни одна из этих функций не является достаточно чувствительной для того, чтобы измерять скорость работы подпрограмм VBA, если подпрограмма не запускается несколько раз подряд.

Большинство современных компьютеров оборудованы таймером высокого разрешения, который обновляется тысячи раз в секунду и доступен через вызов API. Эти вызовы можно реализовать в модуле класса, что позволит другим подпрограммам получать доступ к таймеру высокого разрешения.

Модуль класса HighResTimer

```
Option Explicit

Private Declare Function QueryFrequency Lib "kernel32" _
    Alias "QueryPerformanceFrequency" ( _
    ByRef Frequency As Currency) As Long

Private Declare Function QueryCounter Lib "kernel32" _
    Alias "QueryPerformanceCounter" ( _
    ByRef PerformanceCount As Currency) As Long
```

Обратите внимание, что в файле win32api.txt эти определения показаны с использованием типа данных LARGE_INTEGER, а в предыдущем примере применялся тип Currency. Тип LARGE_INTEGER является 64-разрядным типом данных, который обычно состоит из двух чисел типа long. Тип данных Currency в интерпретаторе VBA также использует 64 двоичных разряда, поэтому его можно применять вместо типа LARGE_INTEGER. Отличием является масштабирование типа данных Currency вниз с коэффициентом 10000 и возможность стандартных арифметических операций над переменными типа Currency.

```
Dim Frequency As Currency
Dim Overhead As Currency
Dim Started As Currency
Dim Stopped As Currency
```

Некоторое время необходимо для завершения работы вызова API. Для получения точных результатов эту задержку стоит учитывать. Эту задержку и частоту таймера можно определить в процедуре обработки события Initialize.

```
Private Sub Class_Initialize()
    Dim Count1 As Currency
    Dim Count2 As Currency

    Call QueryFrequency(Frequency)
```

```
    Call QueryCounter(Count1)
    Call QueryCounter(Count2)
    Overhead = Count2 - Count1
End Sub

Public Sub StartTimer()
    QueryCounter Started
End Sub

Public Sub StopTimer()
    QueryCounter Stopped
End Sub

Public Property Get Elapsed() As Double
    Dim Timer As Currency

    If Stopped = 0 Then
        QueryCounter Timer
    Else
        Timer = Stopped
    End If

    If Frequency > 0 Then
        Elapsed = (Timer - Started - Overhead) / Frequency
    End If
End Property
```

При подсчете прошедшего времени значение таймера и частоты кратно 10000. При делении значений получается результат в секундах.

Класс таймера высокого разрешения может использоваться следующим образом:

```
Sub TestHighResTimer()
    Dim I As Long
    Dim Object As New HighResTimer

    Object.StartTimer

    For I = 1 To 10000
    Next I

    Object.StopTimer

    Debug.Print "10000 итераций потребовали " & Object.Elapsed & " секунд"
End Sub
```

Замораживание диалогового окна UserForm

При работе с диалоговыми окнами UserForm содержимое экрана может изменяться при каждой модификации элементов управления в диалоговом окне, например, при добавлении элемента в список ListBox или включении/отключении элементов управления. Значение свойства Application.ScreenUpdating не оказывает влияния на диалоговые окна UserForm. Класс FreezeForm предоставляет полезный эквивалент этого свойства.

Модуль класса FreezeForm

```
Option Explicit

Private Declare Function FindWindow Lib "user32" _
    Alias "FindWindowA" ( _
```

```
    ByVal ClassИмя As String, _
    ByVal WindowИмя As String) As Long

Private Declare Function LockWindowUpdate Lib "user32" ( _
    ByVal hwndLock As Long) As Long

Public Sub Freeze(Form As UserForm)
    Dim hWnd As Long

    If Val(Application.Version) >= 9 Then
        hWnd = FindWindow("ThunderDFrame", Form.Caption)
    Else
        hWnd = FindWindow("ThunderXFrame", Form.Caption)
    End If
    If hWnd > 0 Then LockWindowUpdate hWnd
End Sub

Public Sub UnFreeze()
    LockWindowUpdate 0
End Sub

Private Sub Class_Terminate()
    UnFreeze
End Sub
```

Для демонстрации создайте новое диалоговое окно UserForm и добавьте список и кнопку. Добавьте следующий код процедуры обработки события Click для кнопки:

```
Private Sub CommandButton1_Click()
    Dim I As Integer
    For I = 1 To 1000
        ListBox1.AddItem "Item " & I
        DoEvents
    Next I
End Sub
```

Строка DoEvents заставляет перерисовываться диалоговое окно UserForm для демонстрации проблемы. В более сложных подпрограммах диалоговые окна UserForm перерисовываются без применения DoEvents. Для предотвращения перерисовки окна можно модифицировать подпрограмму для использования класса FreezeForm.

```
Private Sub CommandButton1_Click()
    Dim Freezer As New FreezeForm
    Freezer.Freeze Me

    Dim I As Integer

    For I = 1 To 1000
        ListBox1.AddItem "Item " & I
        DoEvents
    Next I
End Sub
```

Это намного проще, чем включать несколько вызовов API в каждую функцию. Процедура обработки события Terminate позволяет обеспечить размораживание диалогового окна UserForm при выходе объектной переменной Freezer из области видимости. Такой способ замораживания диалогового окна UserForm может привести к значительному увеличению производительности. Например, при использовании не замороженного диалогового окна заполнение элемента управления ListBox занимает 3,5 секунды. При замораживании диалогового окна эта процедура занимает 1,2 секунды. Данное преиму-

щество необходимо рассматривать с точки зрения взаимодействия с пользователем, так как он при этом может решить, что компьютер завис, если в течение некоторого времени ничего не происходит. Для информирования пользователя о текущем состоянии можно воспользоваться свойством `Application.StatusBar`.

Класс информации о системе

Классическим примером использования модулей классов и вызовов API является предоставление информации о среде Windows, которая недоступна интерпретатору VBA. Следующие свойства — типичные компоненты класса `SysInfo`.

> Обратите внимание, что объявления констант и функций API в этих процедурах должны быть указаны в начале модуля класса. Для ясности эти объявления показаны вместе с соответствующими процедурами.

Получение разрешения экрана (в пикселях)

```
Option Explicit

Private Const SM_CYSCREEN As Long = 1 ' Высота экрана
Private Const SM_CXSCREEN As Long = 0 ' Ширина экрана

Private Declare Function GetSystemMetrics Lib "user32" ( _
    ByVal Index As Long) As Long

Public Property Get ScreenHeight() As Long
    ScreenHeight = GetSystemMetrics(SM_CYSCREEN)
End Property

Public Property Get ScreenWidth() As Long
    ScreenWidth = GetSystemMetrics(SM_CXSCREEN)
End Property
```

Получение глубины цвета (в битах)

```
Private Declare Function GetDC Lib "user32" ( _
    ByVal hwnd As Long) As Long

Private Declare Function GetDeviceCaps Lib "Gdi32" ( _
    ByVal hDC As Long, _
    ByVal Index As Long) As Long

Private Declare Function ReleaseDC Lib "user32" ( _
    ByVal hwnd As Long, _
    ByVal hDC As Long) As Long

Private Const BITSPIXEL = 12

Public Property Get ColourDepth() As Integer
    Dim hDC As Long
    hDC = GetDC(0)
    ColourDepth = GetDeviceCaps(hDC, BITSPIXEL)
    Call ReleaseDC(0, hDC)
End Property
```

Получение ширины пикселя в координатах диалогового окна UserForm

```
Private Declare Function GetDC Lib "user32" ( _
    ByVal hwnd As Long) As Long

Private Declare Function GetDeviceCaps Lib "Gdi32" ( _
    ByVal hDC As Long, _
    ByVal Index As Long) As Long

Private Declare Function ReleaseDC Lib "user32" ( _
    ByVal hwnd As Long, _
    ByVal hDC As Long) As Long

Private Const LOGPIXELSX = 88

Public Property Get PointsPerPixel() As Double
    Dim hDC As Long

    hDC = GetDC(0)

    ' Пункт равен 1/72 дюйма, и LOGPIXELSX возвращает
    ' количество пикселей на логический дюйм, поэтому
    ' разделите это значение
    ' для получения ширины пикселя в координатах диалогового
    ' окна UserForm
    PointsPerPixel = 72 / GetDeviceCaps(hDC, LOGPIXELSX)

    Call ReleaseDC(0, hDC)
End Property
```

Получение регистрационного идентификатора пользователя

```
Private Declare Function GetUserName Lib "advapi32.dll" _
    Alias "GetUserNameA" ( _
    ByVal Buffer As String, _
    ByRef Size As Long) As Long

Public Property Get UserName() As String
    Dim Buffer As String * 255
    Dim Result As Long
    Dim Length As Long

    Length = 255

    Result = GetUserName(Buffer, Length)
    If Length > 0 Then UserName = Left(Buffer, Length - 1)
End Property
```

Получение имени компьютера

```
Private Declare Function GetComputerName Lib "kernel32" _
    Alias "GetComputerNameA" ( _
    ByVal Buffer As String, _
    Size As Long) As Long

Public Property Get ComputerName() As String
    Dim Buffer As String * 255
    Dim Result As Long
    Dim Length As Long

    Length = 255
```

```
    Result = GetComputerName(Buffer, Length)
    If Length > 0 Then ComputerName = Left(Buffer, Length)
End Property
```

Эта подпрограмма может быть протестирована с помощью следующей подпрограммы (в стандартном модуле):

```
Public Sub TestSysInfo()
    Dim Object As New SysInfo
    Debug.Print "Высота экрана = " & Object.ScreenHeight
    Debug.Print "Ширина экрана = " & Object.ScreenWidth
    Debug.Print "Глубина цвета = " & Object.ColourDepth
    Debug.Print "Один пиксель = " & Object.PointsPerPixel
↳ & " пунктов"
    Debug.Print "Имя пользователя = " & Object.UserName
    Debug.Print "Имя компьютера = " & Object.ComputerName
End Sub
```

Модификация стилей диалоговых окон UserForm

Диалоговые окна `UserForm` в Excel не предоставляют встроенного механизма модификации внешнего вида. Единственным выбором является простое всплывающее диалоговое окно с заголовком и кнопкой X для закрытия. Правда, предоставляется возможность выбора между модальным и немодальным диалоговым окном.

Вызовы API позволяют модифицировать диалоговое окно `UserForm` для получения любой комбинации следующих свойств:

- переключение модальности диалогового окна, пока оно отображается на экране;
- включение возможности изменения размера диалогового окна;
- отображение или сокрытие заголовка и названия диалогового окна;
- отображение небольшого заголовка, как на плавающих панелях инструментов;
- отображение собственной пиктограммы;
- отображение пиктограммы в элементе панели задач;
- удаление кнопки [X], закрывающей диалоговое окно;
- добавление стандартных кнопок для разворачивания и сворачивания диалогового окна.

Пример книги, в которой анализируется использование этих свойств, можно получить на сайте `http://www.wrox.com`. Ниже показаны ключевые элементы этого кода.

Свойства окон

Внешний вид и поведение окон управляется свойствами *стилей (style)* и *расширенных стилей (extended style)*. Эти стили являются значениями типа `Long`, в которых каждый бит управляет определенным аспектом внешнего вида окна. Внешний вид окна можно изменить следующим образом:

- использовать функцию `FindWindow` для получения дескриптора диалогового окна `UserForm`;

- прочитать стиль с помощью функции GetWindowLong;
- переключить один или несколько битов стиля;
- заставить окно использовать модифицированный стиль с помощью функции SetWindowLong;
- воспользуйтесь функцией ShowWindow для перерисовки содержимого окна.

Ниже показаны основные константы для каждого бита базового стиля окна:

```
' Стиль для добавления заголовка окна
Private Const WS_CAPTION As Long = &HC00000
' Стиль для добавления системного меню
Private Const WS_SYSMENU As Long = &H80000

' Стиль для добавления фрейма переменного размера
Private Const WS_THICKFRAME As Long = &H40000

' Стиль для добавления кнопки сворачивания диалогового
' окна в заголовок
Private Const WS_MINIMIZEBOX As Long = &H20000

' Стиль для добавления кнопки разворачивания диалогового
' окна в заголовок
Private Const WS_MAXIMIZEBOX As Long = &H10000

' Сбрасывается для отображения пиктограммы в панели задач
Private Const WS_POPUP As Long = &H80000000

' Сбрасывается для отображения пиктограммы в панели задач
Private Const WS_VISIBLE As Long = &H10000000
```

Вот некоторые расширенные стили окна:

```
' Элементы управления при сворачивании окна
Private Const WS_EX_DLGMODALFRAME As Long = &H1

' Окно приложения: отображается в панели задач
Private Const WS_EX_APPWINDOW As Long = &H40000

' Окно инструментов: небольшой заголовок
Private Const WS_EX_TOOLWINDOW As Long = &H80
```

Обратите внимание, что это только подмножество возможных стилей окна. Дополнительная информация доступна в документации MSDN по стилям окон (http://msdn.microsoft.com/library/psdk/winui/windows_2v90.htm) и в файле win32api.txt. Также в этих источниках отображаются значения стилей.

В следующем примере показанные выше процедуры используются для удаления кнопки закрытия диалогового окна UserForm. Этот пример доступен в файле NoCloseButton.xls, который предоставляется на сайте издательства Wrox.

```
Private Const WS_CAPTION As Long = &HC00000
Private Const WS_SYSMENU As Long = &H80000
Private Const WS_THICKFRAME As Long = &H40000
Private Const WS_MINIMIZEBOX As Long = &H20000
Private Const WS_MAXIMIZEBOX As Long = &H10000
Private Const WS_POPUP As Long = &H80000000
Private Const WS_VISIBLE As Long = &H10000000
Private Const WS_EX_DLGMODALFRAME As Long = &H1
Private Const WS_EX_APPWINDOW As Long = &H40000
Private Const WS_EX_TOOLWINDOW As Long = &H80
```

```
Private Declare Function FindWindow Lib "user32" _
    Alias "FindWindowA" ( _
    ByVal ClassИмя As String, _
    ByVal WindowИмя As String) As Long

Private Declare Function GetWindowLong Lib "user32" _
    Alias "GetWindowLongA" ( _
    ByVal hWnd As Long, _
    ByVal Index As Long) As Long

Private Declare Function SetWindowLong Lib "user32" _
    Alias "SetWindowLongA" ( _
    ByVal hWnd As Long, _
    ByVal Index As Long, _
    ByVal NewLong As Long) As Long

Const GWL_STYLE = -16

Private Sub UserForm_Initialize()
    Dim hWnd As Long
    Dim Style As Long

    If Val(Application.Version) >= 9 Then
        hWnd = FindWindow("ThunderDFrame", Me.Caption)
    Else
        hWnd = FindWindow("ThunderXFrame", Me.Caption)
    End If

    Style = GetWindowLong(hWnd, GWL_STYLE)
    Style = (Style And Not WS_SYSMENU)
    SetWindowLong hWnd, GWL_STYLE, Style
End Sub
```

На рис. 16.2 показан результат работы кода.

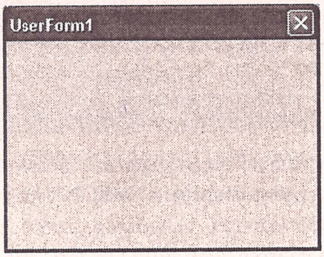


Рис. 16.2. Диалоговое окно UserForm без кнопки закрытия окна

Класс FormChanger

Как было показано ранее в этой главе, вызовы API намного проще использовать через модуль класса. Класс `FormChanger` доступен в файле `FormFun.xls` на сайте издательства Wrox по адресу `http://www.wrox.com/`. В этом классе предыдущий фрагмент кода повторяется для всех битов стиля окна, которые обсуждались в предыдущем разделе. Все биты предоставляются в качестве следующих свойств класса:

- Modal
- Sizeable
- ShowCaption
- SmallCaption
- ShowIcon
- IconPath (для отображения собственной пиктограммы)
- ShowCloseBtn
- ShowMaximizeBtn
- ShowMinimizeBtn
- ShowSysMenu
- ShowTaskBarIcon

Для использования этого класса в собственном диалоговом окне скопируйте весь модуль класса в свой проект и вызовите его из процедуры обработки события Activate, как показано в следующем примере. Этот пример доступен в книге ToolbarForm.xls на сайте http://www.wrox.com.

```
Private Sub UserForm_Activate()
    Dim Object As FormChanger
    Set Object = New FormChanger
    Object.SmallCaption = True
    Object.Sizeable = True
    Set Object.Form = Me
End Sub
```

Диалоговые окна UserForm переменного размера

С выходом Office XP компания Microsoft предоставила возможность изменения размера диалоговых окон Открыть файл (File Open) и Сохранить как (Save As). Окна запоминают положение на экране и размер между сеансами, что значительно увеличивает удобство использования этих окон. Применение показанных в предыдущем разделе вызовов API и модуля классов для сокрытия низкоуровневой реализации позволяет предоставить пользователям те же удобства при работе с диалоговыми окнами UserForm.

Одной из интересных особенностей объекта UserForm является наличие события Resize в то время, как у окна отсутствует свойство, показывающее возможность изменения размера. Теоретически событие Resize никогда не возникает. Как было показано в предыдущем примере, можно предоставить собственное свойство Sizeable. Для этого достаточно установить флажок WS_THICKFRAME. При этом процедура обработки события UserForm_Resize вызывается каждый раз, когда пользователь меняет размер диалогового окна (при перемещении окна на экране это событие не возникает). В ответ на возникновение этого события можно изменить размер и/или положение элементов управления диалогового окна для максимального использования нового размера.

Существует два подхода к изменению размера и/или положения элементов управления: абсолютный и относительный.

Абсолютные изменения

При абсолютном подходе можно создать код, который будет менять размер и положение всех элементов управления диалогового окна относительно новых размеров окна и друг друга. Рассмотрим простое диалоговое окно, на котором присутствует элемент управления `ListBox` и кнопка OK (рис. 16.3).

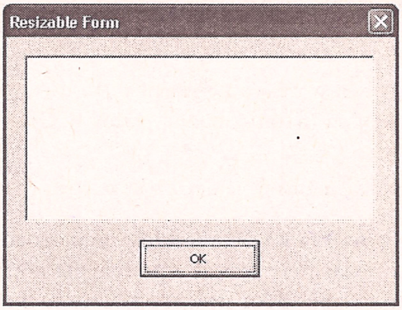


Рис. 16.3. Диалоговое окно, поддерживающее изменение размера

Ниже показан код для изменения размера и перемещения двух элементов управления с использованием абсолютного подхода:

```
Private Sub UserForm_Resize()
    Const Gap = 6
    On Error Resume Next
    CommandButton1.Left = (Me.InsideWidth - CommandButton1.Width) / 2
    CommandButton1.Top = Me.InsideHeight - Gap -
➥ CommandButton1.Height
    ListBox1.Width = Me.InsideWidth - Gap * 4
    ListBox1.Height = CommandButton1.Top - Gap * 2
End Sub
```

Этот код работает, но имеет ряд серьезных недостатков.

- Для каждого элемента управления, который меняет положение или размер, необходимо писать отдельный код. Для больших и сложных диалоговых окон эта задача может стать утомительной. Пример такого кода показан в книге `FormFun.xls`.
- Размер и положение элементов управления часто зависят от размера и положения других элементов управления (например, высота списка `ListBox` зависит от верхней границы кнопки OK).
- При модификации формы через добавление или перемещение элементов управления соответствующие изменения придется вносить в код изменения размера. Например, для добавления кнопки Отмена (Cancel) рядом с кнопкой OK придется добавить код перемещения кнопки Отмена (Cancel) и модифицировать код перемещения кнопки OK.
- Не существует возможности повторного использования кода.

Относительные изменения

При относительном подходе к каждому элементу управления добавляется информация о возможном изменении размера и положения в зависимости от изменения размера и положения диалогового окна UserForm. В том же диалоговом окне два элемента управления поддерживают следующие относительные изменения:

- кнопка OK перемещается вниз на расстояние, равное изменению высоты диалогового окна (это позволяет кнопке OK оставаться в нижней части диалогового окна);
- кнопка OK перемещается в стороны на расстояние, равное половине изменения ширины диалогового окна (это позволяет кнопке OK оставаться посредине диалогового окна);
- высота и ширина списка увеличиваются так же, как и ширина и высота диалогового окна.

Эти операторы можно закодировать в одной строке, указывающей долю изменения каждого из свойств элемента управления (Top, Left, Height и Width). Свойство Tag является удобным местом для хранения такой информации. Это позволяет описывать поведение на этапе проектирования. Если в качестве названий свойств использовать T, L, H и W, а в качестве процента изменения — десятичное число (если изменение составляет 100%, число не указывается), следующие значения свойства Tag можно использовать для описания поведения простого диалогового окна:

```
Tag=HW
Tag=HL0.5
```

При возникновении события UserForm_Resize код рассчитывает изменение высоты и ширины диалогового окна и перебирает все элементы управления, меняя значения Top, Left, Height и Width, указанные в свойстве Tag элемента управления. Выполняющий эту операцию класс CFormResizer показан ниже.

Этот подход обладает несколькими преимуществами:

- поведение каждого элемента управления при изменении размеров диалогового окна определяется на этапе проектирования, когда диалоговое окно можно просматривать одновременно со свойствами элемента управления;
- изменение размера и положения каждого элемента управления описывается независимо от других элементов управления;
- элементы управления можно добавлять, перемещать и удалять без модификации кода изменения размера или изменений в поведении других элементов управления;
- код изменения размеров использует одинаковый подход ко всем элементам управления;
- каждое диалоговое окно UserForm применяет один и тот же код изменения размера, который может быть реализован в виде отдельного модуля класса.

Класс FormResizer

Реализация обработчика изменений размера в виде отдельного модуля класса позволяет обрабатывать любое диалоговое окно UserForm. Для этого в коде диалогового окна достаточно добавить шесть строк для создания и вызова экземпляра класса. Кроме этого, в свойстве Tag каждого элемента управления необходимо описать поведение при изменении размера диалогового окна.

Класс FormResizer предоставляет следующую функциональность:

- разрешает изменение размера диалогового окна;
- устанавливает начальные положение и размер диалогового окна, если диалоговое окно уже отображалось на экране;
- меняет размер и положение всех элементов управления диалогового окна в соответствии со значением свойства Tag;
- сохраняет размер и положение диалогового окна в системном реестре (эта информация будет использоваться при следующем отображении диалогового окна);
- позволяет вызывающему коду передавать имя раздела системного реестра, в котором будут сохраняться параметры диалогового окна;
- запрещает изменение размера диалогового окна, если ни один из элементов управления не настроен на реагирование в этих изменениях;
- останавливает изменение размера, если один из элементов управления достиг левого или верхнего края диалогового окна, или при изменении размеров элемента управления до 0.

Ниже показан код класса FormResizer. Смысл каждого раздела передается в комментариях. Код класса доступен для загрузки на сайте http://www.wrox.com (файл называется FormResizer.xls):

```
Option Explicit

Private Declare Function FindWindow Lib "user32" _
    Alias "FindWindowA" ( _
    ByVal ClassИмя As String, _
    ByVal WindowИмя As String) As Long

Private Declare Function GetWindowLong Lib "user32" _
    Alias "GetWindowLongA" ( _
    ByVal hWnd As Long, _
    ByVal Index As Long) As Long

Private Declare Function SetWindowLong Lib "user32" _
    Alias "SetWindowLongA" ( _
    ByVal hWnd As Long, _
    ByVal Index As Long, _
    ByVal NewLong As Long) As Long

Private Const GWL_STYLE As Long = (-16)
Private Const WS_THICKFRAME As Long = &H40000

Dim FormField As Object
Dim FormHandle As Long
Dim Width As Double
Dim Height As Double
Dim RegistryKeyField As String

Private Sub Class_Initialize()
    RegistryKey = "Excel 2003 Programmers Reference"
End Sub

Public Property Let RegistryKey(ByVal Value As String)
    RegistryKeyField = Value
End Property
```

```
Public Property Get RegistryKey() As String
    RegistryKey = RegistryKeyField
End Property

Public Property Get Form() As Object
    Set Form = FormField
End Property

Public Property Set Form(Value As Object)
    Dim StringSizes As String
    Dim Sizes As Variant
    Dim Style As Long

    Set FormField = Value
    If Val(Application.Version) < 9 Then
        'Excel 97
        FormHandle = FindWindow("ThunderXFrame", FormField.Caption)
    Else
        ' Более новые версии Excel, включая Excel 2003
        FormHandle = FindWindow("ThunderDFrame", FormField.Caption)
    End If

    Style = GetWindowLong(FormHandle, GWL_STYLE)
    Style = Style Or WS_THICKFRAME
    Call SetWindowLong(FormHandle, GWL_STYLE, Style)

    StringSizes = GetSetting(RegistryKey, "Forms", FormField.Имя, "")

    Width = FormField.Width
    Height = FormField.Height

    If StringSizes <> "" Then
        Sizes = Split(StringSizes, ";")
        ReDim Preserve Sizes(0 To 3)
        FormField.Top = Val(Sizes(0))
        FormField.Left = Val(Sizes(1))
        FormField.Height = Val(Sizes(2))
        FormField.Width = Val(Sizes(3))
        FormField.StartUpPosition = 0
    End If

End Property

Public Sub FormResize()
    Dim WidthAdjustment As Double
    Dim HeightAdjustment As Double
    Dim SomeWidthChange As Boolean
    Dim SomeHeightChange As Boolean
    Dim Tag As String
    Dim Size As String
    Dim Control As MSForms.Control

    Static Resizing As Boolean

    If Resizing Then Exit Sub
    Resizing = True
    On Error GoTo Finally

    HeightAdjustment = Form.Height - Height
    WidthAdjustment = Form.Width - Width
```

```
    For Each Control In Form.Controls
        Tag = UCase(Control.Tag)
        If InStr(1, Tag, "T", vbBinaryCompare) Then
            If Control.Top + HeightAdjustment *ResizeFactor(Tag, "T")
➥ <= 0 Then
                Form.Height = Height
            End If

            SomeHeightChange = True
        End If

        If InStr(1, Tag, "L", vbBinaryCompare) Then
            If Control.Left + WidthAdjustment *ResizeFactor(Tag, "L")
➥ <= 0 Then
                Form.Width = Width
            End If

            SomeWidthChange = True
        End If

        If InStr(1, Tag, "H", vbBinaryCompare) Then
            If Control.Height + HeightAdjustment *ResizeFactor(Tag,
➥ "H") <= 0 Then
                Form.Height = Height
            End If

            SomeHeightChange = True
        End If

        If InStr(1, Tag, "W", vbBinaryCompare) Then
            If Control.Width + WidthAdjustment *ResizeFactor(Tag,
➥ "W") <= 0 Then
                Form.Width = Width
            End If

            SomeWidthChange = True
        End If
    Next

    If Not SomeHeightChange Then Form.Height = Height
    If Not SomeWidthChange Then Form.Width = Width

    HeightAdjustment = Form.Height - Height
    WidthAdjustment = Form.Width - Width

    For Each Control In Form.Controls
        With Control
            Tag = UCase(.Tag)

            If InStr(1, Tag, "T", vbBinaryCompare) Then
                .Top = .Top + HeightAdjustment *ResizeFactor(Tag,
➥ "T")
            End If

            If InStr(1, Tag, "L", vbBinaryCompare) Then
                .Left = .Left + WidthAdjustment *ResizeFactor(Tag,
➥ "L")
            End If
```

```
            If InStr(1, Tag, "H", vbBinaryCompare) Then
                .Height = .Height + HeightAdjustment *
↳ ResizeFactor(Tag, "H")
            End If

            If InStr(1, Tag, "W", vbBinaryCompare) Then
                .Width = .Width + WidthAdjustment *ResizeFactor(Tag,
↳ "W")
            End If
        End With
    Next

    Width = Form.Width
    Height = Form.Height

    With Form
        Call SaveSetting(RegistryKey, "Forms", .Имя, Str(.Top) & ";"
↳ & Str(.Left) & ";" & Str(.Height) & ";" & Str(.Width))
    End With

Finally:
    Resizing = False

End Sub

Private Function ResizeFactor(ByVal Tag As String,
↳ ByVal Change As String)
    Dim I As Integer
    Dim D As Double

    I = InStr(1, Tag, Change, vbBinaryCompare)

    If I > 0 Then
        D = Val(Mid$(Tag, I + 1))
        If D = 0 Then D = 1
    End If

    ResizeFactor = D
End Function
```

Использование класса FormResizer

Ниже показан код, который позволяет использовать класс FormResizer в модуле кода диалогового окна UserForm:

```
Dim Resizer As FormResizer

Private Sub UserForm_Initialize()
    Set Resizer = New FormResizer
    Resizer.RegistryKey = "Excel и VBA Справочник программиста"
    Set Resizer.Form = Me
End Sub

Private Sub UserForm_Resize()
    Resizer.FormResize
End Sub

Private Sub btnOK_Click()
    Unload Me
End Sub
```

```
Private Sub UserForm_QueryClose(Cancel As Integer, CloseMode As Integer)
    Resizer.FormResize
End Sub
```

При использовании этого подхода в собственных диалоговых окнах UserForm стоит обратить внимание на несколько моментов:

- изменение размера осуществляется через изменение значения свойств Top, Left, Height и Width элементов управления в ответ на изменение размера диалогового окна UserForm;
- информация для управления изменением размера элементов управления устанавливается в свойстве Tag. Для этого используются символы T, L, H и/или W, после которых указывается множитель изменения (если изменение не составляет 100%);
- множитель изменения должен иметь формат США (в качестве десятичного разделителя должна использоваться точка);
- если ни один из элементов управления не содержит T или H в значении свойства Tag, диалоговое окно не будет поддерживать изменение вертикального размера;
- если ни один из элементов управления не содержит L или W в значении свойства Tag, диалоговое окно не будет поддерживать изменение горизонтального размера;
- минимальный размер диалогового окна определяется первым элементом управления, который достиг верхнего или левого края диалогового окна или получил размер 0 по вертикали или горизонтали;
- предыдущая особенность может использоваться для задания минимального размера диалогового окна с помощью скрытой метки со значением свойства Tag, равным "HW". Размер метки показывает, насколько может быть уменьшено диалоговое окно. Если указать нулевой начальный размер метки, диалоговое окно будет поддерживать только увеличение.

Другие примеры

Вызовы API не обязательно помещать в модули классов, хотя это и очень удобно. В этом разделе рассматривается несколько примеров использования вызовов API в пределах стандартных модулей.

Изменение пиктограммы Excel

При разработке приложения, которое перехватывает управление над всем интерфейсом Excel, можно воспользоваться следующим кодом для предоставления новой пиктограммы Excel:

```
Declare Function FindWindow Lib "user32" _
    Alias "FindWindowA" ( _
    ByVal ClassName As String, _
    ByVal WindowName As String) As Long

Declare Function ExtractIcon Lib "shell32.dll" _
    Alias "ExtractIconA" ( _
    ByVal Instance As Long, _
    ByVal ExeFileName As String, _
    ByVal IconIndex As Long) As Long
```

```
Declare Function SendMessage Lib "user32" _
    Alias "SendMessageA" ( _
    ByVal hWnd As Long, _
    ByVal Message As Long, _
    ByVal wParam As Integer, _
    ByVal lParam As Long) As Long

Const WM_SETICON = &H80

Public Sub SetExcelIcon(ByVal IconPath As String)
    Dim A As Long
    Dim hWnd As Long
    Dim hIcon As Long

    hWnd = FindWindow("XLMAIN", Application.Caption)
    hIcon = ExtractIcon(0, IconPath, 0)

    '1 означает некорректный источник пиктограммы
    If hIcon > 1 Then
        ' Код устанавливает большую (32x32)
        ' и маленькую (16x16) пиктограммы
        Call SendMessage(hWnd, WM_SETICON, True, hIcon)
        Call SendMessage(hWnd, WM_SETICON, False, hIcon)
    End If
End Sub

Public Sub TestExcelIcon()
    Call SetExcelIcon(ThisWorkbook.Path + "\myico.ico")
End Sub
```

Воспроизведение файла .wav

Excel не предоставляет встроенных методов воспроизведения звука. Ниже рассматривается простой вызов API для воспроизведения файлов .wav. Аргумент Flags позволяет включить асинхронное и циклическое воспроизведение звука, хотя в этом примере используется значение 0, приводящее к однократному синхронному воспроизведению звукового файла:

```
Declare Function sndPlaySound Lib "winmm.dll" _
    Alias "sndPlaySoundA" ( _
    ByVal SoundName As String, _
    ByVal Flags As Long) As Long
Sub PlayWav(ByVal WavFileName As String)
    Call sndPlaySound(WavFileName, 0)
End Sub

Sub TestWav()
    Call PlayWav(ThisWorkbook.Path + "\mywav.wav")
End Sub
```

Резюме

Определенные в составе Windows API функции предоставляют полезные и мощные расширения к инструментарию разработчика VBA. В файле win32api.txt содержатся определения VBA для большинства основных функций. Определения остальных функций могут быть получены на основе версий в стиле C, показанных в библиотеке MSDN.

Модули классов позволяют пользователю скрывать определения вызовов API и предоставлять только простые фрагменты функциональности, которые легко повторно использовать в приложениях VBA. В этой главе было рассмотрено несколько примеров классов и подпрограмм. Этого достаточно, чтобы начать использование функций Windows API в собственных приложениях. Показанные примеры позволяют:

- создавать файл в каталоге `TEMP`;
- создавать таймер высокого разрешения;
- блокировать обновление диалогового окна `UserForm`;
- получать информацию о системе;
- модифицировать внешний вид диалогового окна `UserForm`;
- создавать диалоговые окна `UserForm` переменного размера с минимальным объемом кода в модуле диалогового окна;
- менять пиктограмму Excel;
- воспроизводить файл `.wav`.

17

Проблемы интернационализации

Если предполагается, что приложение будет использоваться в других языковых средах, оно должно применять региональные параметры в каждой поддерживаемой языковой версии Windows с любым языком пользовательского интерфейса Excel.

Любые ошибки в приложении, связанные с проблемами интернационализации, не проявляются на компьютере разработчика, если специально не проверять их существование. Но клиенты обнаруживают эти ошибки практически немедленно.

Комбинация региональных параметров и языка пользовательского интерфейса Excel называется "локалью" пользователя. В этой главе показано, как создавать приложения VBA, не зависящие от текущих региональных параметров и языка интерфейса. Для этого будут проанализированы возможности Excel, касающиеся интернационализации, и выделены области Excel, где поддержка интернационализации отсутствует или является ограниченной. В большинстве случаев для каждого ограничения существует обходное решение, но некоторые ограничения настолько сложны, что самым лучшим решением является отказ от использования соответствующей возможности.

Рассмотренные в этой главе правила должны быть включены в стандарты кодирования и использоваться всеми разработчиками в компании. Независимый от локализации код намного проще создавать с самого начала; адаптация существующего кода к различным наборам региональных параметров может оказаться слишком сложной.

Изменение региональных параметров Windows и языка пользовательского интерфейса Office XP

В этой главе потенциальные ошибки будут демонстрироваться на примере следующих трех наборов региональных стандартов:

Параметр	*США*	*Великобритания*
Разделитель целой и дробной части	.	.
Разделитель разрядов	,	,
Формат даты	мм/дд/гггг	дд/мм/гггг
Разделитель даты	/	/
Пример числа: 1234.56	1,234.56	1,234.56
Пример даты: 10 февраля 2004 года	02/10/2004	10/02/2004
Язык Windows и Excel	Английский	Английский
Текстовое представление значения True	True	True

Региональные параметры меняются в диалоговом окне Региональные параметры (Regional Settings) (в Windows XP это диалоговое окно называется Язык и региональные стандарты (Regional and Language Options)), доступном из Панели управления (Control Panel). Язык пользовательского интерфейса Office XP меняется в программе Языковые параметры Microsoft Office, которая предоставляется вместе с языковыми пакетами Office XP. К сожалению, для изменения языка операционной системы придется заново установить ее на компьютер.

> Такие программы виртуализации компьютера, как Connectix Virtual PC или VMWare, позволяют установить тестовую копию операционной системы с интересующими региональными параметрами, не затрагивая основную копию.

При тестировании приложения желательно использовать выдуманные региональные параметры, например: # — в качестве разделителя разрядов, ! — в качестве разделителя целой и дробной части и ГМД — в качестве формата даты. При этом очень легко определить, применяет ли приложение установленные региональные параметры или встроенные принятые по умолчанию. Для полноты проверки стоит воспользоваться другой языковой версией операционной системы Windows.

Обработка региональных параметров и языка интерфейса Windows

В этом разделе рассматривается создание приложений, работающих с различными региональными параметрами и языковыми версиями Windows. Возможность такой работы является абсолютным минимальным требованием к созданию приложений.

Идентификация региональных параметров пользователя и языковой версии Windows

Вся необходимая информация о региональных параметрах и языковой версии операционной системы доступна через свойство `Application.International`. В справочном руководстве перечислены все доступные значения, хотя в реальных приложениях используется только часть параметров. Чаще всего применяются:

- `XlCountryCode` — языковая версия Excel (или активного языка интерфейса Office);
- `XlCountrySetting` — текущее расположение из региональных параметров Windows;
- `XlDateOrder` — один из форматов даты.

Обратите внимание, что не существует константы, определяющей установленную языковую версию Windows (но при необходимости эту информацию можно получить через вызов функции Windows API).

Функции преобразования VBA с точки зрения интернационализации

В интерактивном справочном руководстве рассматривается использование функций преобразования VBA для перехода от одного типа данных к другому. В этом разделе речь идет о применении явного и неявного преобразования типов, а также влиянии региональных параметров на такое преобразование.

Неявное преобразование

Неявное преобразование является наиболее распространенной формой преобразования. При этом интерпретатор VBA вынужден выполнять преобразование данных в наиболее подходящий формат. Вот типичный пример кода, в котором используется неявное преобразование:

```
Public Sub ImplicitConversion()
    Dim MyDate As Date
    MyDate = DateValue("Jan 1, 2004")
    Call MsgBox("Это первый день этого года: " & MyDate)
End Sub
```

При преобразовании числа в строку в Office XP интерпретатор VBA воспользуется региональными параметрами для создания правильно отформатированной даты, числа или текста "True" или "False", соответствующего языку в региональных параметрах. Такой подход оказывается оправданным, если строка должна выдаваться с использованием принятого в данной местности форматирования. Но если код предполагает, что в результате преобразования будет получена строка, формат которой применяется в США, работа кода завершится неудачно. Разработчик, использующий региональные стандарты США, не заметит разницы. Но эта разница станет сразу заметной для клиента.

При создании кода для нескольких версий Excel неявное преобразование типов приводит к еще большему количеству проблем. В предыдущих версиях использовались форматы чисел, соответствовавшие языку Excel, который применялся на этапе выполнения кода (эта информация спрятана глубоко в объектной библиотеке Excel). Такой формат может отличаться от форматов, принятых в США, и от локальных форматов. Кроме это-

го, изменение региональных параметров Windows не оказывало никакого влияния на используемый формат.

Внимательно следите за типами данных, которые возвращаются и используются в функциях Excel и VBA. Например, функция `Application.GetOpenFilename` возвращает тип `Variant`, содержащий значение `False` типа `Boolean`, если пользователь щелкает на кнопке Отмена (Cancel), или строку с названием выделенного файла. Если сохранить этот результат в строковой переменной (тип `String`), значение `False` типа `Boolean` будет преобразовано в строку в соответствии с региональными параметрами Windows. При этом конечный результат преобразования, скорее всего, не будет равен строке "False", с которой сравнивается возвращаемое значение функции.

Для избежания этих проблем найдите типы возвращаемого значения и параметров функции в окне Object Browser (Просмотр объектов) и откажитесь от неявного преобразования через использование совпадающих типов или выполняйте явное преобразование в тип переменной. Следование этим рекомендациям предоставляет (как минимум) три решения для использования функции `Application.GetOpenFilename`.

Типичный код, который может быть запущен в Норвегии:

```
Dim FileName As String
FileName = Application.GetOpenFilename()
If FileName = "False" Then
   ...
```

Если пользователь щелкает на кнопке Отмена (Cancel), функция `GetOpenFilename` возвращает значение `False` типа `Boolean`. На основе региональных параметров Windows приложение Excel выполняет преобразование значения в строку и присваивает результат преобразования переменной. В Норвегии переменная будет содержать строку `"Usann"`. Это значение не совпадает со строкой `"False"`, поэтому приложение решает, что предоставлено действительное имя файла, и рано или поздно аварийно завершит работу.

Решение 1:

```
Dim FileName As Variant
FileName = Application.GetOpenFileName()
If FileName = False Then   ' Сравнение значений одного типа
      ...
```

Решение 2:

```
Dim FileName As Variant
FileName = Application.GetOpenFileName()
If CStr(FileName) = "False" Then ' Явное преобразование с
                                 ' помощью CStr() всегда
                                 ' дает строку Boolean,
                                 ' соответствующую региональным
                                 ' стандартам США
   ...
```

Решение 3:

```
Dim FileName As Variant
FileName = Application.GetOpenFileName()
If TypeName(FileName) = "Boolean" Then  ' Получили значение типа
                                        ' Boolean, значит
                                        ' пользователь щелкнул
                                        ' на кнопке Отмена (Cancel)
   ...
```

Обратите внимание, что в каждом из трех случаев ключевым моментом является сопоставление типа возвращаемого значения функции `GetOpenFileName` (тип `Variant`) с типом собственной переменной. Если в функцию `GetOpenFileName` передается аргумент `MultiSelect:=True`, необходимо использовать последнее решение. Это связано с тем, что переменная `FileName` будет содержать массив имен файлов или значение `False` типа `Boolean`. Попытка сравнения массива со значением `False` или преобразования массива в строку приведет к ошибке времени выполнения.

Строки с датами

При создании кода на языке VBA даты можно записывать в формате `#01/01/2004#`. Очевидно, что эта строка соответствует 1 января 2004 года. Но что означает строка `#02/01/2004#`? Это 2 января или 1 февраля? На самом деле это 1 февраля 2004 года. Связано это с тем, что код VBA использует региональные стандарты США вне зависимости от региональных параметров. В результате приходится применять форматирование строк дат, принятое в США. Если ввести дату в другом формате (`#гггг-мм-дд#`), Excel выполнит автоматическое преобразование в формат `#мм/дд/гггг#`.

Но что если британец или норвежец введет дату в локальном формате (а это обязательно произойдет, когда сроки сдачи работы станут ближе)? Если ввести строку даты в принятом в Норвегии формате **`#02.01.2004#`**, будет выдано сообщение о синтаксической ошибке, которое, как минимум, подскажет о внесенной ошибке. Но если ввести дату, соответствующую региональным стандартам Великобритании (дд/мм/гггг), все станет еще интереснее. Интерпретатор VBA распознает дату и не выдаст сообщения об ошибке, но "обратит внимание", что день и месяц перепутаны местами; интерпретатор автоматически поменяет их. Поэтому ввод дат с 10 января 2004 по 15 января 2004 дает такие результаты:

Введенная строка	*Выводимая строка*	*Реальный смысл*
10/1/2004	10/1/2004	1 октября 2004 года
11/1/2004	11/1/2004	1 ноября 2004 года
12/1/2004	12/1/2004	1 декабря 2004 года
13/1/2004	13/1/2004	13 января 2004 года
14/1/2004	14/1/2004	14 января 2004 года
15/1/2004	15/1/2004	15 января 2004 года

Если такие строки равномерно распределены по всему коду, ошибка останется незаметной.

Намного проще отказаться от использования строк с датами и перейти к использованию функций VBA `DateSerial(Year, Month, Day)` или `DateValue(DateString)`, где строка `DateString` является однозначным определением даты, например, `"Jan 1, 2004"`. Обе функции возвращают соответствующее число типа `Date`.

Функции IsNumeric и IsDate

Эти функции проверяют, можно ли преобразовать строку в число или дату в соответствии с региональными параметрами и языком интерфейса Windows. Всегда используйте эти функции перед преобразованием строки в другой тип данных. Функции `IsBoolean`

не существует. Кроме этого, не существует функций, проверяющих форматирование числа или даты в соответствии с форматами США. Обратите внимание, что функция `IsNumeric` не распознает символ % в конце числа, а `IsDate` — названия дней недели.

Функция CStr

Обычно эта функция используется интерпретатором VBA для неявного преобразования типов данных. Функция выполняет преобразование типа `Variant` в тип `String` в соответствии с региональными параметрами Windows. При преобразовании типа `Date` используется формат `"ShortDate"`, определенный в региональных параметрах Windows. Обратите внимание, что при преобразовании значений типа `Boolean` получается текст `"True"` или `"False"` на английском языке. Результат такого преобразования не зависит от региональных параметров Windows. Сравните это поведение с неявным преобразованием типа `Boolean`, при котором функция `MsgBox "I am " & True` выдает значение `True` на языке, указанном в региональных параметрах Windows (при использовании норвежских региональных параметров в результате вызова функции будет выдано сообщение `"I am Sann"`).

Функции CDbl, CSng, CLng, CInt, CByte, CCur и CDec

Каждая из этих функций выполняет преобразование строкового представления числа в соответствующий тип данных (кроме этого, функции поддерживают преобразование различных типов числовых данных). Строка должна быть отформатирована в соответствии с региональными параметрами Windows. Эти функции не распознают строки с датами и символы `%`.

Функции CDate и DateValue

Эти функции выполняют преобразование строк в тип `Date` (кроме этого, функция `CDate` поддерживает преобразование других типов даты в тип `Date`). Строка должна быть отформатирована в соответствии с региональными параметрами Windows, а для названий месяцев должен использоваться текущий язык интерфейса Windows. Имена дней недели не распознаются (если передать имя дня недели в качестве параметра, выдается сообщение об ошибке `Type Mismatch`). Если в строке не указан год, используется текущий год.

Функция CBool

Функция `CBool` выполняет преобразование строки (или числа) в значение типа `Boolean`. В отличие от остальных функций преобразования `Cxxx`, в данном случае строка должна содержать слова на английском языке `"True"` или `"False"`.

Функция Format

Эта функция выполняет преобразование числа или даты в строку. Формат указывается в качестве аргумента функции. В описании формата должны применяться только символы, используемые в США (`m`, `d`, `s` и т.д.), но в результате работы функции выдается строка, отформатированная в соответствии с региональными параметрами Windows (с правильными десятичными разделителями, разделителями тысяч и дат), а также на языке интерфейса Windows (для названий дней недели и месяцев). Например, в результате работы следующего кода с региональными параметрами США выдается строка

"Friday 01/01/2004". При выполнении этого кода с региональными параметрами Норвегии выдается строка "Fredag 01.01.2004".

```
MsgBox Format(DateSerial(2004, 1, 1), "dddd dd/mm/yyyy")
```

Если строку с форматом числа не указывать, функция Format поведет себя точно так же, как функция CStr (несмотря на то что в интерактивном справочном руководстве указано, что в этом случае функция ведет себя как функция Str), включая странную обработку значений Boolean, в результате которой вызов Format(True) всегда выдает строку "True". Обратите внимание, что порядок компонентов даты не соответствует региональным параметрам Windows, поэтому код должен определить используемый порядок компонентов даты перед созданием строки формата.

Функции FormatCurrency, FormatDateTime, FormatNumber и FormatPercent

Эти функции впервые появились в Excel 2000 и предоставляют ту же функциональность, что и функция Format. Вместо строки формата функции принимают аргументы, определяющие конкретный формат. Значения аргументов соответствуют стандартным вариантам в диалоговом окне Excel, доступном по команде Формат⇨Ячейки⇨Числовой (Format⇨Cells⇨Numbers). Функция Format соответствует переключателю Дополнительный (Custom). С точки зрения интернационализации эти функции демонстрируют такое же поведение, что и функция Format, которая рассматривалась в предыдущем разделе.

Функция Str

Эта функция выполняет преобразование числа, даты или значения типа Boolean в строку, отформатированную в соответствии с региональными стандартами США. Результат работы функции не зависит от региональных параметров Windows, языка пользовательского интерфейса Windows или языковой версии Office. При преобразовании положительного числа слева добавляется пробел. При преобразовании десятичной дроби начальный ноль не добавляется. Следующая функция является расширением функции Str. В ее задачи входит добавление нуля и удаление начального пробела.

Функция NumberToString

Эта функция, как и предыдущая, выполняет преобразование числа, даты или значения типа Boolean в строку, отформатированную в соответствии с региональными стандартами США, но предоставляет дополнительный параметр, который может использоваться для получения строки с помощью функции Excel DATE. Обычно эта функция используется при составлении строк для свойства .Formula.

```
Public Function NumberToString(ByVal Value As Variant, _
    Optional ByVal UseDateFunction As Boolean) As String

    Dim Temp As String
    If TypeName(Value) = "String" Then Exit Function
    If Right(TypeName(Value), 2) = "()" Then Exit Function

    If IsMissing(UseDateFunction) Then UseDateFunction = False

    If UseDateFunction Then
        Temp = "DATE(" & Year(Value) & "," & Month(Value) & "," & _
            Day(Value) & ")"
    Else
        If TypeName(Value) = "Date" Then
            Temp = Format(Value, "mm""/""dd""/""yyyy")
```

```
        Else
            Temp = Trim(Str(Value))
        If Left(Temp, 1) = "." Then Temp = "0" & Temp
            If Left(Temp, 2) = " ." Then Temp = " 0" & Mid(Temp, 2)
        End If
    End If

    NumberToString = Temp
End Function
```

Переменная Value имеет тип Variant и содержит преобразовываемое число, которое может быть:

- числом, которое необходимо преобразовать в строку с использованием региональных стандартов США;
- датой, которую необходимо преобразовать в строку в формате мм/дд/гггг;
- значением типа Boolean, которое необходимо преобразовать в строки "True" или "False".

Переменная UseDateFunction имеет тип Boolean и включает или отключает обработку дат. При установке этой переменной в значение False функция NumberToString возвращает строку даты в формате мм/дд/гггг. При установке этой переменной в значение True функция NumberToString возвращает дату как DATE(гггг, мм, дд).

Функция Val

Эта функция наиболее часто используется для преобразования строк в числовые значения. На самом деле она выполняет преобразование только строк, соответствующих региональным стандартам США. Все остальные функции преобразования строк в числовые значения пытаются преобразовать в число всю строку. При невозможности такого преобразования функции выдают сообщение об ошибке. Но функция Val просматривает значение переменной слева направо, пока не найдет символ, который не может быть проинтерпретирован как часть числа. Многие символы, которые часто встречаются при записи чисел, например символ $ или запятая, могут помешать преобразованию строки в число. Функция Val не поддерживает преобразование строк даты, соответствующих региональным стандартам США.

Кроме этого, функция Val является единственной функцией VBA, принимающей в качестве аргумента только один тип данных. В то время как другие функции принимают тип данных Variant, функция Val поддерживает работу только со строками. Это значит, что любое переданное в функцию Val значение сначала преобразовывается в строку (выполняется неявное преобразование, а значит, зависящее от региональных параметров Windows и языка пользовательского интерфейса Windows), после чего выполняется преобразование с учетом региональных стандартов США.

> Использование функции Val может привести к нежелательным побочным эффектам (также известным, как ошибки), которые очень сложно обнаружить, так как код может прекрасно работать на компьютере разработчика и отказывать на другом компьютере с иными региональными параметрами Windows.

В данном случае переменная myDate имеет тип Date и значение "10 февраля 2004 года". Переменная myDouble имеет тип данных Double и содержит значение 1.234.

Выражение	США	Великобритания	Норвегия
`Val(myDate)`	2	10	10.02 (или 10.2)
`Val(myDbl)`	1.234	1.234	1
`Val(True)`	0 (=False)	0 (=False)	0 (=False)
`Val("SomeText")`	0	0	0
`Val("6 My St.")`	6	6	6

Функция Application.Evaluate

Хотя обычно эта функция не используется для преобразования типов, тем не менее это единственный способ преобразования строки даты, соответствующей региональным стандартам США, в число типа `Date`. Следующие функции представляют собой оболочки, которые используют этот метод объекта `Application`.

Функция IsDateUS

Встроенная функция `IsDate` проверяет строку на соответствие региональным стандартам Windows. Эта функция позволяет определить, содержит ли строка дату, отформатированную в соответствии с региональными стандартами США.

```
Public Function IsDateUS(ByVal aDate As String) As Boolean
    IsDateUS = Not IsError( _
        Application.Evaluate("DATEVALUE(""" & Date & """)"))
End Function
```

Если переменная `aDate` содержит дату в формате, принятом в США, то функция `IsDateUS` возвращает значение `True`; в противном случае функция возвращает значение `False`.

Функция DateValueUS

Функция VBA `DateValue` выполняет преобразование строки даты, соответствующей региональным параметрам Windows, в значение типа `Date`. А функция `DateValueUS` выполняет преобразование строки даты, соответствующей региональным стандартам США, в значение типа `Date`. Если строка имеет другой формат, функция возвращает значение `Error`, которое определяется функцией `IsError`.

```
Public Function DateValueUS(ByVal aDate As String) As Variant
    DateValueUS = Application.Evaluate("DATEVALUE(""" & aDate & """)")
End Function
```

Переменная `aDate` содержит строку даты, отформатированную в соответствии с региональными стандартами США. Функция `DateValueUS` возвращает значение типа `Date` и принимает строку типа `Variant`.

Взаимодействие с Excel

VBA и Excel являются двумя различными программами с совершенно разным воспитанием. VBA говорит на американском английском. Excel тоже говорит на американском английском. Но кроме этого, Excel может говорить на других языках пользователей, если установлены соответствующие параметры Windows и языковые пакеты Office. С другой стороны, VBA очень мало знает о параметрах Windows и еще меньше о языковых пакетах

Office. Следовательно, можно создать нагромождение кода, которое заставит VBA говорить с Excel на языке пользователя, или можно просто позволить VBA и Excel общаться на американском английском. Рекомендуем остановиться на последнем варианте.

К сожалению, большинство новых возможностей Excel не обладает многоязыковой поддержкой. Некоторые поддерживают только американский английский, а есть такие, которые поддерживают только язык пользователя. Первые возможности можно использовать при полном понимании их ограничений. Вторые лучше не применять вообще. Все эти возможности рассматриваются далее в этой главе.

Отправка данных в Excel

Самым лучшим методом загрузки чисел, дат, бинарных значений и строк в ячейки Excel является использование родного формата. Следовательно, показанный ниже код будет работать одинаково вне зависимости от региональных стандартов:

```
Public Sub SendToExcel()
    Dim aDate As Date
    Dim Number As Double
    Dim Bool As Boolean
    Dim Str As String

    aDate = DateSerial(2004, 2, 13)
    Number = 1234.567
    Bool = True
    Str = "Здравствуй мир"

    Range("A1").Value = aDate
    Range("A2").Value = Number
    Range("A3").Value = Bool
    Range("A4").Value = Str
End Sub
```

Между VBA и Excel есть промежуточный слой. При передаче переменной через этот слой от VBA в Excel, Excel пытается интерпретировать значение переменной в соответствии с собственными правилами. Если типы данных VBA и Excel взаимно совместимы, переменная проходит через пограничный слой без изменений.

Проблемы начинаются в тот момент, когда из-за Excel или по собственной инициативе числа, даты или бинарные значения передаются в виде строк. В такой ситуации существует простое решение: всегда перед передачей значения в Excel выполняйте явное преобразование строковых данных в тип данных, который должен храниться в Excel.

Excel может потребовать строкового представления данных в следующих ситуациях:

- создание формулы для ячейки, ряда на диаграмме, условного форматирования или вычисляемого поля сводной таблицы;
- указание формулы `RefersTo` для определенного имени;
- указание критерия `AutoFilter`;
- передача формулы в функцию `ExecuteExcel4Macro`;
- передача формата числа для ячейки, стиля, оси диаграммы или поля сводной таблицы;
- установка формата числа для функции VBA `Format`.

В этих случаях необходимо убедиться, что VBA отправляет в Excel строки, отформатированные в соответствии с региональными стандартами США (нужно использовать английский язык и региональные параметры США). Если строка создается в процессе работы кода, стоит внимательно преобразовать все переменные в строки в соответствии с региональными стандартами США.

Например:

```
Public Sub SetLimit(ByVal Limit As Double)
    ActiveCell.Formula = "=IF(A1<" & Limit & ",1,0)"
End Sub
```

Формула ячейки устанавливается на основе параметра, который предоставляется другой подпрограммой. Обратите внимание, что формула создается в процессе работы кода и при создании строки используются региональные параметры США и английский язык (ключевое слово `IF` и запятая в качестве разделителя списка). При использовании этого кода с различными значениями `Limit` с разными региональными стандартами будут получены следующие результаты:

Значение Limit	*США*	*Великобритания*	*Норвегия*
100	Работает	Работает	Работает
100.23	Работает	Работает	Ошибка времени выполнения 1004

При использовании региональных параметров Норвегии код аварийно завершает работу при любом не целом значении `Limit`. Это связано с тем, что выполняется неявное преобразование числового значения переменной в строковое и результат преобразования зависит от региональных параметров Windows. В результате преобразования в Excel передается следующая строка:

```
=IF(A1<100,23,1,0)
```

Так как в функцию передается четыре параметра, формула не работает. Если изменить код следующим образом:

```
Public Sub SetLimit(ByVal Limit As Double)
    ActiveCell.Formula = "=IF(A1<" & Str(Limit) & ",1,0)"
End Sub
```

формула будет работать правильно, так как вызов `Str` выполнит явное преобразование строки в соответствии с региональными стандартами США.

Если выполнить ту же подпрограмму по отношению к значению типа `Date` вместо `Double`, появится еще одна проблема. Передаваемый в Excel текст (например, для даты 13 февраля 2004 года) будет выглядеть следующим образом:

```
=IF(A1<02/13/2004,1,0)
```

Хотя это действительная формула, Excel воспримет дату в качестве последовательности делений, поэтому формула будет преобразована в

```
=IF(A1<0.000077,1,0)
```

Скорее всего, такое условие никогда не будет истинным. Для обхода этой проблемы необходимо преобразовать тип `Date` в тип `Double`, после чего преобразовать значение в строку:

```
Public Sub SetDateLimit(ByVal Limit As Date)
    ActiveCell.Formula = "=IF(A1<" & Str(CDbl(Limit)) & ",1,0)"
End Sub
```

В результате функция будет работать правильно, хотя станет сложнее для восприятия:

```
=IF(A1<36935,1,0)
```

Для сохранения простоты восприятия даты необходимо преобразовывать в вызовы функции Excel DATE, например:

```
=IF(A1<DATE(2004,2,13),1,0)
```

Кроме этого, можно воспользоваться функцией NumberToString, которая рассматривалась ранее в этой главе. При этом параметр UseDateFunction должен быть установлен в значение True:

```
Public Sub SetDateLimit(ByVal Limit As Date)
    ActiveCell.Formula = "=IF(A1<" & NumberToString(Limit, True)
↳ & ",1,0)"
End Sub
```

Если вызвать последний вариант подпрограммы SetLimit и передать в качестве значения параметра 100.23, можно заметить, что программа Excel выполнила преобразование строки, соответствующей региональным стандартам США, в строку на локальном языке, соответствующую текущим региональным параметрам. Например, в Норвегии этот результат будет выглядеть следующим образом:

```
=HVIS(A1<100,23;1;0)
```

Это преобразование также относится к форматам чисел. При каждой установке формата числа средствами VBA в Excel передается строка формата с использованием символов, принятых в США (например, 'd' для дня, 'm' для месяца и 'y' для года). После применения строки формата к ячейке, стилю или оси диаграммы или использования в функции Format Excel выполняет преобразование этих символов в локальные версии. Например, в результате работы следующего кода в норвежской версии Windows (при вызове команды Формат⇨Ячейки⇨Числовой (Format⇨Cells⇨Number)) будет получена строка dd/mm/eeee:

```
ActiveCell.NumberFormat = "dd/mm/yyyy"
```

Способность Excel выполнять преобразование строк из региональных стандартов США на локальные языки и в региональные форматы позволяет создавать приложения, не зависящие от региональных параметров. Достаточно создавать код с использованием региональных стандартов США и выполнять явное преобразование переменных в соответствующие региональным стандартам США строки перед передачей их в Excel.

Чтение данных из Excel

При чтении значения ячейки с помощью свойства Value возвращаемый Excel тип данных зависит от комбинации значения и форматирования ячейки. Например, число 3,000 может передаваться в VBA как тип Double, тип Currency или тип Date (18 марта 1908 года). Единственная проблема, связанная с интернационализацией, возникает при присвоении значения ячейки непосредственно строковой переменной — в таком случае выполняется неявное преобразование и результат может отличаться от ожидаемого (особенно, если ячейка содержит значение типа Boolean).

Как и в случае с отправкой данных в Excel, при чтении данных выполняется преобразование между региональными стандартами. Это значит, что свойства `.Formula` и `.NumberFormat` возвращаются на английском языке с использованием форматирования чисел и дат, принятого в США, вне зависимости от языка пользовательского интерфейса или региональных параметров операционной системы.

Хотя в большинстве приложений проще работать с формулами и форматами США, иногда возникает необходимость получить именно то представление, которое видит пользователь (с учетом выбранного языка интерфейса и региональных параметров). Для этого можно воспользоваться версиями свойств `xxxLocal`, которые возвращают (и интерпретируют) строки в соответствии с пользовательскими параметрами. Обычно эти свойства применяются для отображения формулы или формата числа в диалоговом окне `UserForm`. Эти версии свойств рассматриваются в следующем разделе.

Правила работы с Excel

Ниже перечислены основные принципы обхода проблем, связанных с интернационализацией листов и кода VBA.

- По возможности передавайте значения в Excel с использованием стандартного форматирования (не выполняйте преобразование дат, чисел и бинарных значений в строковые значения без особой необходимости). Если строковые значения представляют другие типы данных, самостоятельно выполните преобразование до передачи значения в Excel.
- При преобразовании чисел и дат в строки для передачи в Excel (например, при создании критерия для функции `AutoFilter` или строк `.Formula`), всегда выполняйте явное преобразование данных в строки, отформатированные в соответствии с региональными стандартами США. Для этого можно воспользоваться вызовом `Trim(Str(MyNumber))` или функцией `NumberToString`, которая рассматривалась ранее. После этого Excel выполнит корректное преобразование в локальный формат чисел/дат.
- Избегайте использования в коде строкового представления дат (например, `#1/3/2004#`). Лучше воспользуйтесь функцией VBA `DateSerial` или функцией Excel `DATE`, возвращающими однозначный результат.
- По возможности вместо строкового используйте числовое представление даты. Числа намного реже становятся причиной неоднозначности (хотя использование чисел не дает полной гарантии).
- При автоматической генерации формул, которые будут сохранены в ячейках (с использованием свойства `.Formula`), создавайте строку на основе английских названий функций. Excel выполнит автоматическое преобразование имен функций в соответствии с языком пользовательского интерфейса Office.
- При установке формата числа или при вызове функции `Format` применяйте символы форматирования, принятые в США, например `ActiveCell.NumberFormat = "dd mmm yyyy"`. Excel выполнит преобразование в локализованный формат числа.
- При чтении информации с листа с использованием свойств `.Formula`, `.NumberFormat` и так далее Excel будет возвращать английские названия функций и коды форматирования, принятые в США. Возвращаемые значения не зависят от языка пользовательского интерфейса Excel.

Взаимодействие с пользователями

Золотым правилом вывода данных для пользователей или получения данных от пользователей является соблюдение региональных параметров Windows и применение языка пользовательского интерфейса Office. От пользователей нельзя требовать ввода чисел, дат, формул и строк форматирования, соответствующих региональным стандартам США, только потому что это упрощает разработку приложений.

Размер бумаги

Наибольшее раздражение у пользователя вызывает отказ принтера работать с размером бумаги, который применяется в существующих шаблонах приложения. Если шаблоны используются для создания отчетов, всегда меняйте размер бумаги на размер, принятый пользователем по умолчанию. Для определения этого параметра можно создать новую книгу и извлечь размер бумаги из объекта `PageSetup`.

В Excel 2002 было добавлено свойство `Application.MapPaperSize`, которое позволяет автоматически переключаться между стандартными размерами бумаги, принятыми в различных странах (например, размер Letter в США эквивалентен A4 в Великобритании). Если свойство `Application.MapPaperSize` установлено в значение `True`, Excel автоматически будет следить за используемым размером бумаги.

Вывод данных

Excel хорошо справляется с отображением листов в соответствии с региональными параметрами и языком пользовательского интерфейса. Но при выводе данных через диалоговые окна `UserForm` или диалоговые листы форматирование приходится выполнять самостоятельно.

Как было показано ранее, по умолчанию преобразование чисел и дат в строки в Excel выполняется в соответствии с региональными параметрами Windows. Это значит, что можно написать следующий код и при этом быть уверенным, что Excel выведет информацию в правильном формате.

```
TextBoxNumber.Text = 3.14159
```

Но с неявным преобразованием типов (в показанном фрагменте кода выполняется неявное преобразование числа с плавающей точкой в строковое значение) связано две проблемы.

- Для преобразования дат применяется принятый по умолчанию формат `"ShortDate"`, в котором для представления года не выделяется четыре цифры и отсутствует представление времени. Для вывода года с помощью четырех цифр и информации о времени можно воспользоваться функцией `FormatDate`, показанной далее в этой главе. Но в диалоговых окнах `UserForm` можно воспользоваться более однозначным форматом, например форматом "ммм дд, гггг", который применяется на протяжении этой книги.
- В версиях Excel до Excel 97 региональные параметры Windows не влияли на выбор форматирования. При создании приложений для более старых версий Excel на правильное поведение рассчитывать нельзя.

В этом случае существует простое решение — воспользуйтесь функцией `Format`. Эта функция заставляет VBA выполнять преобразование числа в отформатированную в соответствии с региональными стандартами строку. Кроме этого, функция `Format` работает во всех версиях Excel, начиная с Excel 5.0.

```
TextBoxNumber.Text = Format(3.14159)
```

Интерпретация данных

Скорее всего, пользователи будут вводить даты и числа в соответствии с региональными параметрами, а код должен выполнять соответствующие проверки и, возможно, выдавать осмысленные сообщения об ошибках. Это значит, что придется использовать функции преобразования `Cxxx`, а также функции проверки `IsNumeric` и `IsDate`.

К сожалению, эти функции обладают собственными недостатками (например неспособностью распознавать символ `%` в конце числа), требующими обходных решений. Простым решением является использование показанных в конце этой главы функций `WinToNum` и `WinToDate`. Эти функции выполняют проверку действительности и преобразование, а также выдают соответствующие сообщения об ошибках. Код проверки ввода для диалогового окна `UserForm` обычно вызывается из процедуры обработки события `Click` для кнопки OK. Код может быть реализован следующим образом:

```
Private Sub CommandButtonOK_Click()
    Dim result As Double

    If WinToNum(TextBoxNumber.Text, result, True) Then
        Sheet1.Range("A1").Value = result
    Else
        TextBoxNumber.SetFocus
        Exit Sub

    End If
    Me.Hide

End Sub
```

Свойства xxxLocal

До этого момента было сказано, что с Excel придется взаимодействовать с использованием английских названий функций и форматов, принятых в США. Ниже приводится альтернативная ситуация, при которой код взаимодействует с пользователем на языке пользователя. Для этого применяются соответствующие региональные стандарты. Возникает вопрос: как приложение может считать введенные пользователем данные (формат числа или формулу) и передавать их непосредственно Excel или отображать формулу в окне сообщения на языке пользователя?

Компания Microsoft осознала подобное требование и предоставила локализованные версии большинства необходимых функций. Функции имеют те же имена, что и их эквиваленты в США. При этом в конец имени функции добавляется суффикс `"Local"` (например, `FormulaLocal`, `NumberFormatLocal` и т.д.). При использовании этих функций Excel не выполняет автоматическое преобразование форматов и языков. Считываемый и записываемый текст ничем не отличается от того, который видит пользователь. Практически все функции, которые возвращают строки или принимают строковые аргументы, имеют локальные эквиваленты. В следующей таблице перечислены все функции и объекты, к которым они применяются.

Применяется к	*Эти версии функций принимают и возвращают строки на английском языке, соответствующие региональным стандартам США*	*Эти версии функций принимают и возвращают строки на языке пользовательского интерфейса Office (или Windows), отформатированные в соответствии с локальными региональными параметрами*
Преобразование строк/чисел	`Str`	`CStr`
Преобразование строк/чисел	`Val`	`CDbl` и т.д.
Имя, Стиль, Командная панель	`.Name`	`.NameLocal`
Диапазон, Ряд на диаграмме	`.Formula`	`.FormulaLocal`
Диапазон, Ряд на диаграмме	`.FormulaR1C1`	`.FormulaR1C1Local`
Диапазон, Стиль, Метка данных на диаграмме, Метка осей на диаграмме	`.NumberFormat`	`.NumberFormatLocal`
Диапазон	`.Address`	`.AddressLocal`
Диапазон	`.AddressR1C1`	`.AddressR1C1Local`
Определенное имя	`.RefersTo`	`.RefersToLocal`
Определенное имя	`.RefersToR1C1`	`.RefersToR1C1Local`
Определенное имя	`.Category`	`.CategoryLocal`

Правила работы с пользователями

- При преобразовании числа или даты в текстовую строку для последующего отображения пользователям или присвоения свойствам `.Caption` и `.Text` элементов управления, выполняйте явное преобразование чисел и дат в соответствии с региональными параметрами Windows. Для этого можно воспользоваться функциями `Format` или `CStr`.
- При преобразовании дат в строки Excel не меняет порядок компонентов даты, поэтому вызов функции `Format(MyDate, "дд/мм/гггг")` всегда будет выдавать дату в порядке ДМГ (хотя при этом будет использоваться правильный разделитель даты). Воспользуйтесь вызовом `Application.International(xlDateOrder)` для определения правильного порядка даты. Этот вызов применяется в функции `FormatDate`, показанной в конце этой главы. Кроме этого, можно воспользоваться одним из стандартных форматов даты, например `ShortDate`.
- По возможности используйте форматы даты, не зависящие от региональных параметров, например `Format(MyDate, "ммм дд, гггг")`. В этом случае Excel будет выводить имена месяцев в соответствии с языком региональных параметров Windows.

- При интерпретации введенных пользователем строки даты или строки числа воспользуйтесь функциями `CDate` и `CDbl`, позволяющими преобразовывать строку в дату или число. Для разбора строки эти функции используют региональные параметры Windows. Обратите внимание, что функция `CDbl` не распознает символ `%` в конце числа.
- Всегда проверяйте допустимость введенных пользователем дат и чисел до попытки преобразования. Пример проверки приводится в функциях `WinToNum` и `WinToDate`, показанных далее в этой главе.
- При выводе информации об объектах Excel используйте свойства `xxxLocal` (если такие свойства существуют). Это позволит получить данные на языке пользователя с сохранением форматирования.
- Если свойствам объектов Excel присваиваются введенные пользователем значения, применяйте свойства `xxxLocal`. (Предполагается, что пользователь вводит значения на своем родном языке в принятом в данной местности формате).

Возможности интернационализации в Excel 2003

В Excel 2003 в диалоговое окно Сервис⇨Параметры (Tools⇨Options) была добавлена вкладка Международные (International). В этой вкладке пользователь может указать символы, которые Excel должна применять в качестве десятичных разделителей и разделителей тысяч, переопределяя региональные параметры Windows. Для изменения и получения значений этих параметров можно воспользоваться свойствами `Application.ThousandSeparator`, `Application.DecimalSeparator` и `Application.UseSystemSeparators`.

Эти свойства позволяют распечатывать, сохранять (в текстовом формате) или публиковать книги с использованием локального формата чисел. При этом можно изменить значения этих свойств, сохранить (в текстовом формате) или опубликовать книгу для другой страны и восстановить начальные значения параметров. К сожалению, компания Microsoft не предоставила возможности переопределения других региональных параметров Windows — например, порядка компонентов даты, разделителя компонентов даты или представления отрицательных чисел.

Одной из проблем в работе параметров Международные (International) является невосприимчивость строк форматирования чисел в вызовах функции листа =TEXT, поэтому после изменения параметра (в результате работы кода или через пользовательский интерфейс) все ячейки с функцией =TEXT получат неправильный формат чисел. Решение этой проблемы рассматривается далее в этой главе.

Добавление параметров Международные (International) отрицательно сказалось и на разработчиках. Проблема заключается в том, что хотя эти параметры оказывают влияние на все свойства Excel `xxxLocal` и функции (включая `Application.International`), они полностью игнорируются интерпретатором VBA.

Вот несколько примеров, которые позволяют понять глубину проблемы:

- функция VBA `Format`, которая используется практически при каждом выводе числа для пользователя, игнорирует значения этих параметров, а значит отображаемый текст будет отформатирован в соответствии с региональными параметра-

ми Windows, а не в соответствии с установленными параметрами Международные (International);

- если пользователь вводит числа в диалоговое окно UserForm и элемент управления InputBox с применением переопределенных разделителей, введенные числа не будут восприниматься такими функциями преобразования, как IsNumeric и CDbl. В результате будет выдаваться сообщение об ошибке TypeMismatch.

Единственный способ обойти эту проблему — самостоятельное переключение между региональными параметрами Windows и переопределенными разделителями перед выводом чисел для пользователя и непосредственно после получения чисел от пользователя. Для этого можно воспользоваться следующими функциями:

```
Public Function WRSToOverride(ByVal Number As String) As String

    Dim WRS As String
    Dim Thousand As String
    Dim aDecimal As String
    Dim XLThousand As String
    Dim XLDecimal As String

    If Val(Application.Version) >= 10 Then
        If Not Application.UseSystemSeparators Then
            WRS = Format(1000, "#,##0.00")
            Thousand = Mid(WRS, 2, 1)
            aDecimal = Mid(WRS, 6, 1)
            XLThousand = Application.ThousandsSeparator
            XLDecimal = Application.aDecimalSeparator
            Number = Replace(Number, Thousand, vbTab)
            Number = Replace(Number, aDecimal, XLDecimal)
            Number = Replace(Number, vbTab, XLThousand)
        End If
    End If
    WRSToOverride = Number
End Function
```

Функция WRSToOverride выполняет преобразование между форматами чисел, принятыми в региональных параметрах Windows и Excel. Она возвращает строку, отформатированную в соответствии с переопределенными разделителями в Excel. Переменная Number содержит строковое представление числа, отформатированное в соответствии с региональными параметрами Windows.

```
Public Function OverrideToWRS(ByVal Number As String) As String

    Dim WRS As String
    Dim WRSThousand As String
    Dim WRSDecimal As String
    Dim XLThousand As String
    Dim XLDecimal As String

    If Val(Application.Version) >= 10 Then
        If Not Application.UseSystemSeparators Then
            WRS = Format$(1000, "#,##0.00")
            WRSThousand = Mid$(WRS, 2, 1)
            WRSDecimal = Mid$(WRS, 6, 1)
            XLThousand = Application.ThousandsSeparator
            XLDecimal = Application.DecimalSeparator
            Number = Replace(Number, XLThousand, vbTab)
            Number = Replace(Number, XLDecimal, WRSDecimal)
```

```
            Number = Replace(Number, vbTab, WRSThousand)
        End If
    End If
    OverrideToWRS = Number

End Function
```

Функция `OverrideToWRS` выполняет преобразование между форматами региональных параметров Windows и Excel. Она возвращает строку, отформатированную в соответствии с региональными параметрами Windows. Переменная `Number` содержит строковое представление числа, отформатированное с использованием переопределенных разделителей Excel.

Последняя проблема возникает при взаимодействии с пользователем, так как для этого необходимо применять знакомые пользователю форматы чисел. После добавления возможности переопределения региональных параметров Windows в Excel предоставляется третий набор разделителей, с которыми придется работать пользователю и разработчикам. При этом разработчик полностью зависит от того, запомнил ли пользователь текущий переопределенный разделитель, а это могут быть не те разделители, которые пользователь привык видеть (определенные в региональных параметрах Windows).

Рекомендуется, чтобы приложение проверяло значение свойства `Application.UseSystemSeparators` и сравнивало это свойство со значением `True`. Пользователю должно выдаваться сообщение с предложением отключить переопределенные разделители и установить эти параметры с помощью Панели управления (Control Panel).

```
If Application.UseSystemSeparators Then
    MsgBox "Устанавливайте форматирование чисел
↳ через Панель управления"
    Application.UseSystemSeparators = False
End If
```

Возможности, не следующие общим правилам

Функции `xxxLocal`, которые рассматривались в предыдущем разделе, были добавлены в Excel в процессе переноса функций XLM в VBA с выходом Excel 5.0. Такие функции существуют для большинства распространенных функций, которые будут использоваться разработчиком. Но в процессе оригинального преобразования были упущены несколько возможностей. Кроме этого, с тех пор в Excel некоторые возможности были добавлены совершенно без учета вопросов интернационализации.

В данном разделе приводятся советы по выходу из лабиринта неоднородности, плохого проектирования и упущений, скрытых среди возможностей Excel 2003. В этой таблице показаны методы, свойства и функции Excel, которые чувствительны к региональным параметрам пользователя, но их поведение не соответствует описанным ранее правилам.

Применяется к	*Версия США*	*Локализованная версия*
Открытие текстового файла	`OpenText`	`OpenText`
Сохранение текстового файла	`SaveAs`	`SaveAs`
Приложение	`.ShowDataForm`	`.ShowDataForm`

Применяется к	*Версия США*	*Локализованная версия*
Лист/Диапазон		.Paste/.PasteSpecial
Вычисляемые поля и элементы сводной диаграммы	.Formula	
Условные форматы		.Formula
Таблицы запросов (Web-запросы)		.Refresh
Функции листов		=TEXT
Диапазон	.Value	
Диапазон	.FormulaArray	
Диапазон	.AutoFilter	.AutoFilter
Диапазон		.AdvancedFilter
Приложение	.Evaluate	
Приложение	.ConvertFormula	
Приложение	.ExecuteExcel4Macro	

К счастью, для большинства этих проблем существуют решения. Но есть несколько возможностей, которых стоит полностью избегать.

Использование функции OpenText

Функция VBA `Workbooks.OpenText` является аналогом открытия текстового файла в Excel с помощью команды Файл⇨Открыть (File⇨Open). Функция открывает текстовый файл, разбирает его на числа, даты, бинарные значения и строки и сохраняет результат в ячейках листа. Более подробно эта функция рассматривается в других разделах книги. В данном случае интерес вызывает метод, с помощью которого Excel выполняет разбор файла с данными (а также изменения в этом методе с выходом нескольких последних версий).

В Excel 5, если текстовый файл открывался через пользовательский интерфейс, он разбирался в соответствии с региональными параметрами Windows. Если файл открывался в результате работы кода, он разбирался в соответствии с региональными стандартами США. В Excel 97 это поведение изменилось и региональные параметры стали использоваться как в пользовательском интерфейсе, так и в процессе работы кода. К сожалению, это означало отсутствие гарантий правильного открытия текстового файла, отформатированного в соответствии с региональными стандартами США. С выходом Excel 5 представилась возможность описания порядка компонентов даты для каждого столбца в отдельности. Эта возможность хорошо подходит для работы с числовым представлением дат (например, 01/02/2004).

В Excel 2000 в мастере Text Import Wizard была предоставлена кнопка Дополнительно (Advanced) и связанные с этой кнопкой параметры `DecimalSeparator` и `ThousandsSeparator` метода `OpenText`. Эти параметры позволяют указать разделители, которые Excel будет использовать для идентификации чисел. Неприятно то, что точно так же нельзя указывать общий порядок компонентов даты.

```
Public Sub OpenTextTest()

   Dim FileName As String
```

```
    FileName = ThisWorkbook.Path & "\Data.txt"
    Call Workbooks.OpenText(FileName:=FileName, _
        DataType:=xlDelimited, Tab:=True, _
        DecimalSeparator:=",", ThousandsSeparator:=".")

End Sub
```

С выходом Excel 2000 компания Microsoft решила проблемы форматирования чисел, а с выходом Excel 2002 были решены проблемы с названиями месяцев и дней. Кроме этого, были предоставлены более простые альтернативы определения локализованных форматов и форматов США для дат, чисел и других данных.

До выхода Excel 2003 метод OpenText распознавал названия месяцев и дней только в соответствии с региональными параметрами Windows. При этом для всех дат, порядок компонентов которых отличался от МДГ, требовалось указывать порядок компонентов даты. В Excel 2002 у метода OpenText появился параметр Local, который позволяет сообщать методу, что импортируемый текстовый файл отформатирован в соответствии с региональными стандартами США или в соответствии с текущими региональными параметрами Windows:

- если параметр Local установлен в значение True, Excel распознает числа, даты и названия месяцев и дней в соответствии с региональными параметрами Windows (и переопределенными десятичными разделителями и разделителями тысяч, если они установлены);
- если параметр Local установлен в значение False, Excel распознает числа, даты и названия месяцев и дней в соответствии с региональными стандартами США.

В любом случае дополнительные параметры DecimalSeparator, ThousandsSeparator и FieldInfo могут использоваться для дальнейшего уточнения спецификации (и переопределения принятых по умолчанию значений параметраLocal).

Функция SaveAs

Функция VBA Workbook.SaveAs является эквивалентом сохранения текстового файла в Excel с помощью команды Файл⇨Сохранить как (File⇨Save As) и выбора текстового формата.

Во всех версиях Excel до Excel 2002 в результате вызова этой функции создавался текстовый файл, отформатированный в соответствии с региональными стандартами США.

Метод SaveAs принимает тот же параметр Local, что и метод OpenText, который рассматривался ранее. Использование этого параметра позволяет при необходимости получить локализованную версию файла. Обратите внимание, что если в ячейке сохранить дату в локализованном формате (такой формат числа начинается с указателя регионального кода, например, [$-814] для Норвегии), это форматирование будет сохранено в текстовом файле, даже если файл соответствует региональным стандартам США.

```
Public Sub SaveText()
    Dim FileName As String
    FileName = ThisWorkbook.Path & "\Data.txt"

    Call ActiveWorkbook.SaveAs(FileName, xlText, local:=True)
End Sub
```

Подпрограмма ShowDataForm

Использование метода ActiveSheet.ShowDataForm может подвести разработчика к одной из самых опасных проблем, связанных с интернационализацией. Подпрограмма ShowDataForm является эквивалентом команды Данные⇨Форма (Data⇨Form). В результате вызова этого метода отображается стандартное диалоговое окно, в котором пользователь может вводить или менять данные в списке Excel или в базе данных. При вызове этой подпрограммы из меню выводимые даты и числа отформатированы и интерпретируются в соответствии с региональными параметрами Windows. Такое поведение полностью соответствует правилам взаимодействия с пользователями, описанным ранее в этой главе.

При вызове метода ActiveSheet.ShowDataForm даты и числа отображаются в диалоговом окне с использованием региональных стандартов США, а интерпретируются в соответствии с региональными параметрами Windows. Таким образом, если дата 10 февраля 2004 года отображается на листе в порядке дд/мм/гггг в виде 10/02/2004, Excel покажет эту дату в формате 2/10/2004. После изменения даты на 11 февраля 2004 года (2/11/2004) Excel сохранит дату как 2 ноября 2004 года. Точно так же при использовании норвежских форматов чисел число 1-разделитель целой и дробной части-234 будет показано в виде 1.234. После изменения этого числа на 1.235 Excel сохранит 1235 — число, в тысячу раз большее.

К счастью, если приходится работать с Excel 97 и более поздними версиями, для этой проблемы существует решение. Вместо использования метода ShowDataForm можно выделить первую ячейку диапазона и вызвать пункт меню Данные⇨Форма (Data⇨Form) самостоятельно.

```
Public Sub ShowForm()
   ActiveSheet.Range("A1").Select
   ' 860 идентификатор объекта CommandBarControl
   ' пункта меню Данные⇨Форма
   RunMenu 860
End Sub
```

Показанная ниже подпрограмма RunMenu вызывает интересующий пункт меню так же, как при щелчке мышью. В данном случае диалоговое окно ведет себя правильно.

Подпрограмма RunMenu

Подпрограмма RunMenu вызывает пункт меню, эмулируя щелчок мышью. Для этого в подпрограмму необходимо передать идентификатор CommandBar.Control (например, идентификатор пункта меню Данные⇨Форма (Data⇨Form) равен 860).

```
Public Sub RunMenu(ByVal MenuId As Long)

   Dim Control As CommandBarButton

   On Error Resume Next
   With Application.CommandBars.Add
       .Controls.Add(ID:=MenuId).Execute
   .Delete
   End With
End Sub
```

Параметр MenuId содержит идентификатор пункта меню, который необходимо запустить.

Вставка текста

При вставке текста в Excel из других приложений текст разбирается в соответствии с региональными параметрами Windows. Способа сообщить Excel информацию о формате чисел и дат, а также о языке вставляемого текста не существует. Единственным решением является использование объекта DataObject для получения текста из буфера обмена, самостоятельный разбор текста средствами VBA и запись результата разбора на лист. В следующем примере предполагается, что в буфере обмена содержится число, отформатированное в соответствии с региональными стандартами США.

```
Public Sub ParsePastedNumber()
    Dim obj As DataObject
    Set obj = New DataObject
    obj.GetFromClipboard
    Dim text As String
    ActiveCell.Value = Val(obj.GetText)
End Sub
```

Вычисляемые поля и элементы сводной таблицы, а также формулы условного форматирования

Свойство .Formula диапазона или последовательности на диаграмме принимает и возвращает строки формул на английском языке с использованием форматов чисел, принятых в США. Существует эквивалентное свойство .FormulaLocal, возвращающее и принимающее строки формул в том виде, в котором они отображаются на экране (на языке пользовательского интерфейса Office и с использованием форматов чисел, определенных в региональных параметрах Windows).

Вычисляемые поля и элементы сводной таблицы, а также элементы условного форматирования тоже предоставляют свойство .Formula, но для этих объектов оно принимает и возвращает строки формул в том виде, в котором они отображаются пользователю. Другими словами, это свойство ведет себя точно так же, как свойство .FormulaLocal объекта Range. Это значит, что для установки формулы в одном из этих объектов придется создавать формулу на языке пользовательского интерфейса Office в соответствии с региональными параметрами Windows.

Решением этой проблемы является использование собственных свойств .Formula и .FormulaLocal ячейки для преобразования форматов, как показано в следующей функции ConvertFormulaLocale.

Функция ConvertFormulaLocale

Эта функция выполняет преобразование строки формулы между региональными стандартами США и локальными языками и форматами:

```
Public Function ConvertFormulaLocale( _
   ByVal Formula As String, ByVal USToLocal As Boolean) As String

   On Error GoTo Catch
   With ThisWorkbook.Worksheets(1).Range("IU1")
       If USToLocal Then
           .Formula = Formula
           ConvertFormulaLocale = .FormulaLocal
       Else
           .FormulaLocal = Formula
           ConvertFormulaLocale = .Formula
       End If
```

```
        .ClearContents
    End With

Catch:

End Function
```

Параметр `Formula` содержит текст формулы для преобразования. Для преобразования из региональных стандартов США в локализованные форматы параметр `USToLocal` должен быть установлен в значение `True`. При обратном преобразовании параметр `USToLocal` должен быть установлен в значение `False`.

Web-запросы

Хотя концепция Web-запросов выглядит привлекательно, ее реализация совершенно не считается с вопросами интернационализации. При разборе текста Web-страницы в Excel все числа и даты интерпретируются в соответствии с текущими региональными параметрами Windows. Это значит, что при открытии европейской Web-страницы в США или при открытии Web-страницы из США в Европе, числа и даты будут интерпретироваться неправильно. Например, если на Web-странице содержится текст 1.1, на компьютере с норвежскими региональными параметрами этот текст будет проинтерпретирован как 1 января.

Свойство `WebDisableRecognition` объекта `QueryTable` может использоваться для отключения интерпретации чисел как дат. Если формат вывода Web-страницы известен заранее, для обеспечения правильной интерпретации текста можно воспользоваться переопределенными разделителями.

В многонациональных приложениях следует соблюдать осторожность при использовании Web-запросов. Определенную степень безопасности может обеспечить следующий подход:

- установить свойство `Application.UseSystemSeparators` в значение `False`;
- установить свойства `Application.DecimalSeparator` и `Application.ThousandsSeparator` в значения, соответствующие разделителям на Web-странице;
- выполнить запрос, установив свойство `WebDisableRecognition` в значение `True`;
- установить свойства `Application.DecimalSeparator`, `Application.ThousandsSeparator` и `Application.UseSystemSeparators` в оригинальные значения.

Использование функции листа TEXT

Функция листа TEXT выполняет преобразование числа в строку в соответствии с указанным форматом. В строке формата используются символы форматирования, определенные в региональных параметрах Windows (или в переопределении параметров интернационализации в Excel). Таким образом, в результате вызова `=TEXT(NOW(),"dd/mm/yyyy")` в норвежской версии Windows будет получена строка "01/02/yyyy", так как в этой версии в качестве символа года Excel распознает только символ `'e'`.

При открытии файла на другой платформе Excel не выполняет преобразования символов форматирования чисел. Решением этой проблемы является создание определенного имени, которое будет считывать формат числа из определенной ячейки. После этого такое

определение будет использоваться в функции TEXT. Например, если в ячейке A1 используется формат даты, который применяется в пределах остальной части листа, выберите Вставка⇨Имя⇨Присвоить (Insert⇨Name⇨Define) и определите имя следующим образом:

```
Name:             DateFormat
Refers To:        =GET.CELL(7,$A$1)
```

После этого в любом месте листа можно будет использовать вызовы вида =TEXT(Now(),DateFormat). Функция GET.CELL является макрофункцией Excel 4. Excel допускает применение этой функции с определенными именами, но не в ячейках листа. Этот вызов эквивалентен вызову функции листа =CELL, но является более мощным. В данном примере параметр 7 заставляет функцию GET.CELL возвращать строку формата числа, которая используется в указанной ячейке.

> Обратите внимание, что в предыдущих версиях Excel некоторые пользователи сталкивались с общими ошибками защиты при копировании ячеек с форматом DateFormat в другие листы и книги.

Функции XLM документированы в файле XLMACR8.HLP, который доступен на сайте компании Microsoft по адресу http://support.microsoft.com/support/kb/articles/Q143/4/66.asp

Свойства Range.Value и Range.FormulaArray

Эти свойства объекта Range нарушают установленные правила в том понимании, что не предоставляют локализованные эквиваленты. Свойства принимают и возвращают строки в формате, соответствующем региональным стандартам США. Для преобразования формул можно воспользоваться показанной выше функцией ConvertFormulaLocale.

Метод Range.AutoFilter

Метод AutoFilter объекта Range имеет очень интересное поведение. Он принимает строки, которые используются в качестве критерия фильтрации, поэтому разработчик должен быть знаком с поведением этого метода при обработке строк. Строка критериев состоит из оператора (=, >, <, >= и т.д.), после которого указывается значение. Если оператор не указан, предполагается, что используется оператор '='.

Главной проблемой при использовании оператора '=' является выполнение текстового сравнения, в то время как при использовании других операторов выполняется сравнение значений. Это приводит к проблемам при попытке найти точные совпадения дат и чисел. Применяя оператор '=', Excel выполняет сравнение с текстом, который отображается на экране, то есть, сравнивает критерий с отформатированным числом. Так как отображаемый текст меняется при переходе на другую локализованную версию Windows, создание критерия для точного совпадения на любой локализованной версии Windows невозможно.

Существует решение, позволяющее обойти эту проблему. При использовании любого другого критерия фильтрации Excel выполняет правильное преобразование и интерпретирует строку критерия в соответствии с региональными стандартами США. Таким образом, критерий поиска ">=02/01/2004" позволит найти все даты после 1 февраля 2004 года и будет работать во всех локализованных версиях Windows. Эта особенность позволяет находить точную дату с использованием двух критериев AutoFilter. Показанный ниже код находит точное соответствие дате 1 февраля 2004 года и работает во всех локализованных версиях Windows:

```
Range("A1:D200").AutoFilter 2, ">=02/01/2004", xlAnd, "<=02/01/2004"
```

Метод Range.AdvancedFilter

Метод AdvancedFilter соблюдает правила, но иногда следование правилам может оказаться нежелательным. Критерий фильтрации вводится в диапазон критерия на листе. Как и в методе AutoFilter, в строке критерия указывается оператор и значение. Обратите внимание, что при использовании оператора '=' метод AdvancedFilter выполняет корректное сравнение и в этом отношении отличается от метода AutoFilter.

Так как строка критерия используется только в пределах Excel, она должна быть отформатирована в соответствии с текущими региональными параметрами Windows. В результате при сравнении чисел и дат возникает проблема. В США критерию поиска расширенного фильтра ">1.234" соответствуют все числа больше 1,234. Но в Норвегии этому же критерию будут соответствовать числа больше 1234. В США критерию поиска ">02/03/2004" будут соответствовать даты после 3 февраля. А в Европе этому же критерию соответствуют даты после 2 марта.

Единственное решение — создание строки критерия средствами VBA перед вызовом метода AdvancedFilter или использование вычисляемой строки критерия, которая создается с помощью метода =TEXT, как было показано ранее. Вместо критерия ">=02/03/2004" для поиска дат после 3 февраля 2004 года можно воспользоваться следующей формулой:

```
=">="&TEXT(DATE(2004,2,3),DateFormat)
```

В данном случае DateFormat является определенным именем, возвращающим локальный формат даты. Если дата представляет собой целое число (не содержит компонент времени), можно воспользоваться строкой критерия ">=36194". Пользователю остается догадаться, что 36194 означает дату 3 февраля 2004 года.

Использование функций Application.Evaluate, Application.ConvertFormula и Application.ExecuteExcel4Macro

Эти функции следуют установленным правилам и принимают строки, отформатированные в соответствии с региональными стандартами США. Но эти строки не предоставляют локализованных эквивалентов. Для передачи формулы, которую пользователь ввел в диалоговом окне UserForm (или преобразования формулы между относительными и абсолютными ссылками на диапазоны ячеек), методам Application.Evaluate и Application.ConvertFormula придется преобразовать ее в соответствии с региональными стандартами США.

Функция Application.ExecuteExcel4Macro позволяет выполнять функции в стиле XLM. Одним из самых распространенных применений этой функции является вызов функции XLM PAGE.SETUP, которая работает намного быстрее соответствующего эквивалента из VBA. Эта функция принимает большое количество параметров, включая строки, числа и бинарные значения. Не забудьте выполнить явное преобразование всех параметров в соответствии с региональными стандартами США и сопротивляйтесь соблазну сократить объем кода за счет отказа от вызова функции Str для каждого параметра.

Обработка языковых параметров Office XP

Одним из основных преимуществ версий после Office 2000 является использование одного набора исполнимых файлов со сменными языковыми модулями (до этого использовались разные языковые версии бинарных пакетов с собственными ошибками). Это

позволило пользователю самостоятельно выбирать язык пользовательского интерфейса, файлов справочного руководства и других компонентов пакета. На самом деле, если один компьютер поддерживает работу нескольких пользователей, каждый пользователь может выбрать собственный язык пользовательского интерфейса Office.

Разработчики приложений Excel должны уважать выбор пользователя и стараться предоставить пользовательский интерфейс, который максимально соответствует выбранному пользователем языку.

Откуда извлекается текст?

Существует три фактора, которые вместе определяют текст, выводимый в пользовательском интерфейсе Office.

Хранилище региональных параметров

Расположение региональных параметров указывается на первой вкладке (Региональные параметры (Regional Options)) в окне Панели управления (Control Panel) Язык и региональные параметры (Regional and Language Options). В региональных параметрах определяются:

- ❑ имена дней и месяцев, отображаемые в ячейках Excel при использовании длинных форматов файлов;
- ❑ имена дней и месяцев, которые возвращаются в результате вызова функции VBA `Format`;
- ❑ имена месяцев, которые распознаются при использовании функции VBA `CDate` или при непосредственном вводе дат в Excel;
- ❑ имена месяцев, распознаваемые мастером импорта текста и методом `OpenText` (при установке параметра `Local` в значение `True`);
- ❑ символы форматов чисел, используемые в функции листа `=TEXT`;
- ❑ текст, полученный в результате неявного преобразования значения типа `Boolean` в строковое значение, например `"I am " & True`.

Языковые параметры пользовательского интерфейса Office

Язык пользовательского интерфейса Office устанавливается с помощью утилиты Пуск⇨Программы⇨Инструменты Microsoft Office⇨Языковые параметры Microsoft Office 2003 (Start⇨Program Files⇨Microsoft Office Tools⇨Microsoft Office 2003 Language Settings), которая устанавливается вместе с Office. Кроме этого, в этой утилите определяются:

- ❑ текст, отображаемый в меню и диалоговых окнах Excel;
- ❑ текст стандартных кнопок в окнах сообщений Excel;
- ❑ текст встроенных функций листов Excel;
- ❑ текст значений типа `Boolean`, отображаемый в ячейках Excel;
- ❑ текст значений типа `Boolean`, распознаваемый мастером импорта текста, методом VBA `OpenText` и пользовательским интерфейсом Excel;
- ❑ принятые по умолчанию имена листов в новой книге;
- ❑ локализованные имена панелей меню.

Языковая версия Windows

Языковая версия Windows определяет:

- текст стандартных кнопок в функции VBA `MsgBox` (при использовании констант `vbMsgBoxStyles`). Таким образом, текст кнопок на встроенных сообщениях Excel соответствует языку пользовательского интерфейса Office, а текст кнопок в созданных разработчиком сообщениях соответствует языку Windows. Обратите внимание, что единственным способом определения текущего языка пользовательского интерфейса Windows является применение вызова Windows API.

Существуют некоторые элементы Office XP, которые полностью ориентированы на английский язык (США) и не меняются вместе с изменением языка Windows, региональных параметров или языка пользовательского интерфейса Office, например:

- текст в результате явного преобразования значений типа Boolean в строковые значения, то есть, вызовы `Str(True)`, `CStr(True)` и `Format(True)` всегда возвращают значение `"True"`. Следовательно, единственным способом преобразования значений типа `Boolean` в строковые значения, отображаемые в Excel, является ввод этого значения в ячейку и чтение значения ячейки через свойство `.FormulaLocal`;
- текст значения типа `Boolean`, который распознается функцией `CBool`.

Идентификация языковых параметров пользовательского интерфейса Office

Первым этапом по созданию многоязыкового приложения является идентификация пользовательских параметров. Для идентификации языка, выбранного в региональных параметрах Windows можно воспользоваться вызовом `Application.International (xlCountrySetting)`. В результате вызова этого метода возвращается число, которое примерно соответствует коду региона в телефонной системе. Например, код 1 соответствует США, код 44 — Великобритании, а 47 — Норвегии. Вызов `Application.International(xlCountryCode)` можно использовать для получения языка пользовательского интерфейса. Коды языков определяются по такой же схеме. Этот метод хорошо работал в предыдущих версиях Excel, которые существовали примерно для 30 языков.

Начиная с Office 2000 все немного изменилось. Выделением языковой конфигурации в отдельный пакет компания Microsoft относительно просто обеспечила поддержку намного большего количества языков. В окне Object Browser (Просмотр объектов) найдите константу `msoLanguageID`, определенную в объектной библиотеке Office и обратите внимание на перечисление 180 языков и диалектов.

Следующее значение используется для точного определения языка пользовательского интерфейса Office. После этого можно принять решение о возможности отображения приложения на этом языке, похожем языке или на языке, принятом по умолчанию (как показано в следующем фрагменте кода):

```
LanguageID = Application.LanguageSettings.LanguageID(msoLanguageIDUI)
```

Создание многоязыковых приложений

При разработке многоязыкового приложения приходится находить компромисс с учетом нескольких факторов:

- времени и стоимости разработки приложения;
- времени и стоимости перевода приложения;
- времени и стоимости тестирования переведенного приложения;
- увеличения продаж за счет переведенной версии;
- повышенной простоты использования, а значит, снижения стоимости поддержки;
- требования многоязыковой поддержки;
- необходимости выбора между созданием языковой версии приложения или использованием встраиваемых языковых пакетов.

Кроме этого, необходимо принять решение об объемах перевода и наборе поддерживаемых языков, а именно:

- ничего не переводить;
- переводить только упаковку и рекламную документацию;
- проектировать код для работы в многоязыковой среде;
- переводить пользовательский интерфейс (меню, диалоговые окна, экраны и сообщения);
- переводить файлы справочного руководства, примеры и учебники;
- настраивать приложение для работы в конкретном регионе;
- поддерживать только языки с направлением письма слева направо;
- поддерживать языки с направлением письма справа налево;
- поддерживать языки с двухбайтовыми наборами символов, например японский.

Решение о полноте перевода зависит от пользователей, бюджета и доступности переводчиков.

Рекомендуемый подход

Скорее всего, создание приложения Excel, поддерживающего все 180 языков Office, не оправдано в экономическом смысле, но вот создание приложения, поддерживающего несколько распространенных языков может стать разумным вложением средств. Конечно, окончательное решение зависит от пользователей и от предпочтительности поддержки нового языка.

Предложенный здесь подход позволяет создавать приложения, поддерживающие несколько языков и предоставляющие пользователю возможность переключения между установленными языками или на язык пользовательского интерфейса Office. Разработка приложений ведется на английском языке, после чего выполняется перевод на один или два языка в зависимости от целевой аудитории. Перевод на другие языки целесообразен только в случае гарантированного спроса или в случае, если перевод является обязательным требованием.

Как хранятся строковые ресурсы

При создании многоязыковых приложений в код нельзя вставлять текстовые строки, которые будут выдаваться пользователю. Эти строки должны извлекаться из строковых ресурсов. Самой простой формой строковых ресурсов является таблица на листе. Текстовым элементам необходимо присвоить уникальные идентификаторы и сохранить их на листе; по одному идентификатору на строку и по одному столбцу для каждого поддерживаемого языка. После этого можно найти интересующий идентификатор и вернуть строку из столбца соответствующего языка. Для этого можно воспользоваться функцией VLOOKUP.

Это же придется проделать для всех пунктов меню, содержимого листов и элементов управления UserForm. Ниже показан простой пример кода, в котором предполагается существование листа Language, содержащего таблицу поиска Translation. Кроме этого предполагается, что существует открытая переменная, идентифицирующая столбец, из которого необходимо читать текст. Обычно такая переменная устанавливается в специальном окне с параметрами.

Обратите внимание, что нижеприведенный код не обеспечивает максимальной производительности и рассматривается только в качестве примера. Более быстрая (и более сложная) подпрограмма считывала бы весь столбец с идентификаторами и текст выбранного языка в два простых массива VBA и работала бы с этими данными до выбора другого языка (при этом соответствующий столбец также считывался бы в массив).

```
Public LanguageColumn As Integer

Public Sub Test2()
    LanguageColumn = 2
    Call MsgBox(GetText(1001))
End Sub

Public Function GetText(ByVal TextId As Long) As String
    Dim Test As Variant
    Static LanguageTable As Range

    If LanguageTable Is Nothing Then
        Set LanguageTable = ThisWorkbook.Worksheets("Language") _
        .Range("Translation")
    End If

    If LanguageColumn < 2 Then LanguageColumn = 2
        Test = Application.VLookup(TextId, LanguageTable, LanguageColumn)

    If Not IsError(Test) Then GetText = Test
End Function
```

Многие сообщения создаются на этапе выполнения. Например, может существовать код, проверяющий вхождение числа в определенный диапазон значений:

```
Dim Message As String
    Message = _
        "Число должно быть больше, чем " & CStr(Min) & _
        "и меньше, чем " & CStr(Max) & "."
    If Value <= Min Or Value >= Max Then
        Call MsgBox(Message)
    End If
```

Это значит, что на листе ресурсов придется хранить две текстовые строки с разными идентификаторами, что неэффективно и усложняет перевод. В данном случае, при использовании одного варианта перевода строки, максимальное значение всегда будет на-

ходиться в конце приложения, что допустимо не во всех языках. Более правильный подход предполагает хранение комбинированной строки с символами шаблонов для двух чисел. Шаблоны можно заменять числами на этапе выполнения (для этого можно воспользоваться функцией ReplaceHolders, показанной в конце главы):

```
Dim Message As String
Message = _
    "Число должно быть больше %0 и меньше %1."

If Value <= Min Or Value >= Max Then
    Call MsgBox(ReplaceHolders(Message, CStr(Min), CStr(Max)))
End If
```

Переводчик (который не всегда понимает смысл программы) может правильно создать предложение, вставив значения в соответствующие позиции.

Работа в многоязыковой среде

Обратите внимание на несколько полезных советов по работе в многоязыковой среде.

Оставляйте свободное место

В большинстве языков слова состоят из большего количества символов, чем в английском языке. При проектировании диалоговых окон UserForm и листов необходимо оставлять достаточное количество места для размещения неанглийского текста на элементах управления и в ячейках. Хорошим правилом считается создание элементов управления, которые в 1.5 раза больше соответствующего английского текста.

Использование объектов Excel

Имена создаваемых объектов Excel часто зависят от языка пользовательского интерфейса Office. Например, при создании пустой книги с помощью метода Workbooks.Add книга не всегда будет называться "BookN", а первый лист не всегда будет называться "Sheet1". Если применить немецкую версию пользовательского интерфейса, книга и лист будут называться "MappeN" и "Tabelle1" соответственно. Вместо обращения к объектам по имени создайте ссылку на объект и проинициализируйте ее в момент создания. Созданную ссылку на объект можно будет использовать в любом месте кода:

```
Dim aWorkbook As Workbook
Dim aWorksheet As Worksheet

Set aWorkbook = Workbooks.Add
Set aWorksheet = Wkb.Worksheets(1)
```

Работа с элементами управления CommandBarControl также усложняется. Например, в меню **Сервис** (Tools) можно добавить собственный пункт меню, а в англоязычной среде написать следующий код:

```
Public Sub AddHelloButton()
    Dim Tools As CommandBarPopup
    Dim Control As CommandBarButton

    Set Tools = Application.CommandBars("Worksheet Menu Bar") _
        .Controls("Tools")
        Set Control = Tools.CommandBar.Controls.Add(msoControlButton)
    Control.Caption = "Hello"
    Control.OnAction = "Hello"
End Sub
```

Если используется отличный от английского язык пользовательского интерфейса, этот код не сработает. Хотя Excel распознает английские названия панелей меню, английские названия элементов управления не распознаются. В этом примере не распознается раскрывающееся меню Сервис (Tools). Правильным решением будет применение идентификатора элементов управления CommandBar.Control. Для поиска элементов управления можно использовать метод FindControl. Всплывающее меню Сервис (Tools) имеет идентификатор 30007.

```
Sub AddHelloButton2()
    Dim Tools As CommandBarPopup
    Dim Control As CommandBarButton

    Set Tools = Application.CommandBars("Worksheet Menu Bar") _
        .FindControl(ID:=30007)

    Set Control = Tools.CommandBar.Controls.Add(msoControlButton)

    Control.Caption = "Hello"
    Control.OnAction = "Hello"
End Sub
```

При определенных наборах региональных параметров в некоторых объектных библиотеках существует еще одна проблема, связанная с именами панелей меню (например с панелями меню в голландской версии редактора VBE). В этих локализованных версиях ошибочно локализованы имена панелей меню (все панели меню должны иметь имена на английском языке). Единственным надежным методом работы с панелями меню является отказ от использования имен в коде. Вместо этого необходимо применять функцию FindControl. Этот подход оказывается немного сложнее, так как один и тот же элемент управления может встречаться на разных панелях, а функция FindControl не всегда возвращает интересующий элемент управления. Большинство разработчиков используют английские имена панелей меню.

> В главе 26 будет приведена подпрограмма, которая выводит список всех панелей и элементов управления, а также их имена и идентификаторы. Ян Карел Питерсе (Jan Karel Pieterse) создал книгу с большим количеством переводов панелей. Этот файл называется xlMenuFunDict и доступен по адресу http://www.BMSLtd.ie/MVP.

Использование функции SendKeys

В большинстве случаев использования функции SendKeys стоит избегать. Чаще всего она применяется для отправки нажатий клавиш компонентам Excel для активизации меню или перемещения по диалоговому окну. Функция использует комбинации клавиш пунктов меню и элементов управления в диалоговых окнах таким же образом, как пользователь может применять комбинации <Alt+клавиша> для управления Excel с клавиатуры. Скорее всего, в локализованной версии Excel будут использоваться другие комбинации клавиш, а значит результат применения стандартных комбинаций в функции SendKeys предсказать невозможно.

Например, вызов SendKeys "%DB" в англоязычной версии Excel приведет к открытию диалогового окна Итоги (Subtotals). В версии Excel с немецким языком пользовательского интерфейса этот вызов приведет к завершению работы Excel. Вместо применения SendKeys для выбора пунктов меню можно воспользоваться подпрограммой RunMenu, показанной ранее в этой главе. В этой подпрограмме меню выбирается на основе идентификатора объекта CommandBarControl.

Правила разработки многоязыковых приложений

- ❑ На раннем этапе анализа приложения выберите интересующий уровень многоязыковой поддержки и обеспечивайте этот уровень в процессе разработки.
- ❑ Не включайте в код текстовые строки. Всегда извлекайте строки из таблицы.
- ❑ Никогда не создавайте предложение путем составления отдельных текстовых строк, так как на другом языке предложение может иметь иной порядок слов. Вместо этого используйте символы шаблонов и подставляйте конкретные значения на этапе выполнения.
- ❑ При создании диалоговых окон `UserForm` делайте элементы управления больше, чем необходимо для английского текста; в большинстве языков слова длиннее, чем в английском.
- ❑ Не пытайтесь угадать имя, которое будет присвоено объектам Excel при создании. Например, при создании книги первый лист не всегда будет называться "Sheet1".
- ❑ Не ссылайтесь на элементы управления по названию. Хотя на сами панели можно ссылаться по имени на английском языке, для обращения к пунктам меню необходима ссылка по идентификатору (для встроенных пунктов) или по значению свойства `Tag` (для собственных пунктов меню).
- ❑ Не используйте функцию `SendTags`.

Полезные функции

Кроме уже показанных дополнительных функций, например `RunMenu` и `IsDateUS`, ниже приводятся еще те дополнительные функции, которые оказываются полезными при разработке многоязыковых приложений. Обратите внимание, что код совместим со всеми версиями Excel, от 5.0 до 2003, а значит не использует более новых конструкций VBA (например, объявление типа необязательных параметров).

Реализация функции WinToNum

Функция `WinToNum` определяет, является ли строка числом, отформатированным в соответствии с региональными параметрами Windows. Если это так, строка преобразовывается в значение типа `Double`. Для индикации результата преобразования функция возвращает значение `True` или `False`. Кроме этого, она может выводить сообщение об ошибке. Эту функцию стоит использовать в качестве вспомогательной при проверке чисел, введенных пользователем. Её применение показано в предыдущем разделе "Взаимодействие с пользователями".

Обратите внимание, что если пользователь переопределил разделитель целой и дробной части и разделитель разрядов, которые указываются в региональных параметрах Windows, строка должна быть преобразована с помощью функции `OverrideToWRS` и только после этого передаваться функции `WinToNum`.

```
Private Sub Show(ByVal message As String)
   Const mask As String = _
       "Это сообщение не распознается как " & message & _
       " в соответствии с региональными параметрами Windows"

   Call MsgBox(mask, vbOKOnly)
End Sub
```

```
Public Function WinToNum(ByVal winString As String, _
    ByRef result As Double, Optional ByVal ShowMessage As Boolean) As Boo-
lean

    Dim Fraction As Double

    winString = Trim(winString)
    Fraction = 1

    If IsMissing(ShowMessage) Then ShowMessage = True
    If winString = "." Then winString = "0"
    If winString = "" Then winString = "0"

    If InStr(1, winString, "%") > 0 Then
        Fraction = Fraction / 100
        winString = Application.Substitute(winString, "%", "")
    End If

    If IsNumeric(winString) Then
        result = CDbl(winString) * Fraction
        WinToNum = True
    Else
        If ShowMessage Then Call Show("number")
        result = 0
        WinToNum = False
    End If

End Function
```

В переменной WinString содержится строка для преобразования, а в переменной Result — преобразованное число. Если строка не является числом или пустая, переменная Result устанавливается в нулевое значение. Параметр ShowMessage является необязательным. Для вывода сообщения об ошибке установите его в значение True (или вообще не устанавливайте). Для отключения сообщения об ошибке установите параметр в значение False.

Реализация функции WinToDate

Функция WinToDate предоставляет функциональность, подобную функциональности WinToNum, но выполняет преобразование дат, а не чисел:

```
Public Function WinToDate(ByVal winString As String, _
    ByRef result As Double, Optional ByVal ShowMessage As Boolean) _
    As Boolean

    If IsMissing(ShowMessage) Then ShowMessage = True

    If winString = "" Then
        result = 0
        WinToDate = True

    ElseIf IsDate(winString) Then
        result = CDbl(CDate(winString))
        WinToDate = True
    Else
        If ShowMessage Then Call Show("date")
        result = 0
        WinToDate = False
    End If
End Function
```

Аргумент `WinString` содержит строку для преобразования. В аргумент `Result` заносится преобразованное число. Если строка не является датой или является пустой строкой, аргумент `Result` устанавливается в нулевое значение. Аргумент `ShowMessage` является необязательным. Для вывода сообщения об ошибке этот аргумент должен быть установлен в значение `True` (или должен отсутствовать). Для подавления вывода сообщения установите этот аргумент в значение `False`.

Реализация функции FormatDate

Эта функция выполняет форматирование даты в соответствии с региональными параметрами Windows. Для указания года используются четыре цифры. Кроме этого, предоставляется возможность добавления строки времени:

```
Public Function FormatDate(ByVal aDate As Date, _
    Optional ByVal IncludeTime As Boolean) As String

    Dim DateString As String

    If IsMissing(IncludeTime) Then IncludeTime = False

    Select Case Application.International(xlDateOrder)
        Case 0
            aDate = Format$(aDate, "mm/dd/yyyy")
        Case 1
            Date = Format$(aDate, "dd/mm/yyyy")
        Case 2
            Date = Format$(aDate, "yyyy/mm/dd")
    End Select

    If IncludeTime Then DateString = aDate & " " & _
       Format$(aDate, "hh:mm:ss")

    FormatDate = DateString

End Function
```

Аргумент `aDate` содержит численное представление даты, а `IncludeTime` является необязательным аргументом, который можно установить в значение `True`, что позволит добавить к результату строку времени.

Реализация функции ReplaceHolders

Функция `ReplaceHolders` выполняет подстановку символов шаблонов. Значения для подстановки предоставляются в качестве аргументов:

```
Public Function ReplaceHolders(ByVal Str As String, _
                ParamArray Args() As Variant) As String

    Dim I As Integer

    For I = UBound(Replace) To LBound(Replace) Step 1
        Str = WorksheetFunction.Substitute(Str, "%" & I, _
            Replace(I - LBound(Args)))
    Next
    ReplaceHolders = Str
End Function
```

Аргумент `Str` содержит текст, в котором выполняется замещение шаблонов. Массив `Replace` содержит список заменяемых шаблонов.

Резюме

В Excel можно создавать приложения, которые будут работать в любой версии Excel и на любом из 180 языков пользовательского интерфейса, хотя создание таких приложений не всегда оправдано экономически.

Если количество пользователей приложения ограничено и есть возможность принудительно назначить язык пользовательского интерфейса и региональных параметров Windows, приложение можно создавать, не задумываясь о вопросах интернационализации. Даже в этой ситуации стоит выработать привычку создавать код, независимый от региональных параметров. Требование независимости от региональных параметров должно фигурировать в документах по анализу и проектированию, а также по стандартам написания кода. Намного проще и дешевле сразу написать код, независимый от региональных параметров, чем перерабатывать существующее приложение.

Приложение, как минимум, должно работать вне зависимости от региональных параметров Windows, языка пользовательского интерфейса Windows или языка пользовательского интерфейса Office, даже если с помощью команды Сервис⇨Параметры⇨Международные (Tools⇨Options⇨International) были установлены собственные разделители тысяч и сотен.

Следующие возможности Excel не следуют правилам и должны использоваться с особой осторожностью:

- функция `OpenText`;
- использование метода `SaveAs` для сохранения текстового файла;
- метод `ShowDataForm`;
- вставка текста из других приложений;
- все формы свойства `.Formula`;
- `<range>.Value`;
- `<диапазон>.FormulaArray`;
- `<диапазон>.AutoFilter`;
- `<диапазон>.AdvancedFilter`;
- функция листа `=TEXT`;
- `Application.Evaluate`;
- `Application.ConvertFormula`;
- `Application.ExecuteExcel4Macro`;
- Web-запросы.

Кроме этого, существуют некоторые возможности Excel, которых стоит избегать любой ценой:

- функция `SendKeys`;
- использование значений `True` и `False` в импортируемых текстовых файлах.

При создании многоязыковых решений стоит тестировать реализацию на каждом поддерживаемом языке.

18

Книги и листы

В этой главе речь пойдет о программном управлении книгами и листами. Здесь будут рассмотрены важные фундаментальные процедуры, которые помогают эффективно управлять двумя базовыми объектами Excel. В частности, будут рассматриваться коллекции книг и листов, а также код, необходимый для сохранения и перезаписи этих объектов в процессе изменения состояния.

Использование коллекции Workbooks

Коллекция `Workbooks` является свойством объекта `Application`. Через эту коллекцию можно получить доступ ко всем открытым книгам каждого из экземпляров Excel. Как и все коллекции, коллекция `Workbooks` проектировалась в качестве хранилища для объектов `Workbook`. Кроме этого, как и во всех коллекциях, в ней поддерживается добавление и удаление элементов. Рассмотрим некоторые программные операции, которые поддерживаются коллекцией `Workbooks` и объектами `Workbook`.

Как подсказывает название, коллекция является просто хранилищем набора объектов. Разные коллекции практически ничем не отличаются друг от друга: объекты добавляются, подсчитываются и удаляются. Различаются типы объектов, хранящихся в пределах коллекции, и именно объекты предоставляют интересующую функциональность. Таким образом, все показанные ниже методики на самом деле являются операциями над объектами `Workbook`.

Создание книги

Базовая идея в основе хорошей объектной модели подразумевает предоставление однородных и интуитивных возможностей. Например, если есть набор сущностей, то вполне понятно, что должна быть возможность добавления сущностей в этот набор. На

самом деле, интуитивно понятно, что все коллекции должны поддерживать одинаковое поведение. Простым методом добавления книги в приложение является вызов метода Add коллекции Application.Workbooks:

Вызов метода Workbooks.Add добавляет новый экземпляр класса Workbook в экземпляр приложения Excel и делает эту книгу активной. Кроме этого, объявляется объект Workbook и результат работы метода Add присваивается переменной типа Workbook. В итоге переменная выступает в роли удобной ссылки на новый объектWorkbook.

Сохранение активной книги

Одной из ожидаемых операций над книгой является сохранение. Книгу можно записать в файл, предположив, что недавно добавленная книга является активной. Кроме этого, можно воспользоваться объектной переменной, которой присваивается результат работы метода Workbooks.Add. Ниже показаны операторы добавления книги и сохранения активной книги в файле:

```
Application.Workbooks.Add
Call Application.ActiveWorkbook.SaveAs("temp.xls")
```

В результате работы этого кода создается новый объект Workbook. Ссылка на созданный объект не сохраняется. Вместо этого для взаимодействия с объектом используется свойство Application.ActiveWorkbook. Еще одним подходом является обращение к коллекции для получения конкретного экземпляра класса Workbook или сохранение ссылки на созданный объект. Любая из этих ссылок может использоваться в вызове метода SaveAs. Ниже показана реализация каждого из подходов:

```
' Сохранение копии, полученной по номеру из коллекции Workbooks
Application.Workbooks.Add
Call Application.ActiveWorkbook.SaveAs("temp1.xls")
Application.Workbooks("temp1.xls").SaveAs ("copy of temp1.xls")

' Использование сохраненной ссылки на добавленный объект
Dim W As Workbook
Set W = Application.Workbooks.Add
Call W.SaveAs("temp2.xls")
```

В первом примере показано, как создавать новую книгу и сохранять ее, получая ссылку через свойство ActiveWorkbook. После этого ссылка на книгу извлекается из коллекции по имени книги и с помощью полученной ссылки создается резервная копия. Во втором примере показано, как сохранять ссылку на созданный экземпляр класса Workbook и вызывать метод SaveAs этого экземпляра. Обе операции приводят к одному и тому же результату. Важно обратить внимание, что существует различие между операциями над объектом коллекции Workbooks и отдельным объектом Workbook. Можно заметить, что постоянно используется метод Add коллекции Workbooks и метод SaveAs отдельного объекта Workbook. Следовательно, можно сказать, что коллекция поддерживает метод Add и обращение к элементам коллекции, а объект Workbook поддерживает поведение SaveAs. Это касается всех объектов: если известен класс объекта, на который ссылается код, можно определить доступное поведение этих объектов.

Активизация книги

Более правильным способом создания ссылки на новую книгу является использование возвращаемого значения метода Add для создания объектной переменной. При этом проще обращаться к созданным книгам и отслеживать временные книги без их сохранения на диске.

```
Sub NewWorkbooks()
    Dim Workbook1 As Workbook
    Dim Workbook2 As Workbook

    Set Workbook1 = Workbooks.Add
    Set Workbook2 = Workbooks.Add
    Workbook1.Activate
End Sub
```

Метод Add позволяет указывать шаблон для создания новой книги. В качестве шаблона не обязательно использовать файл шаблона с расширением `.xlt`. Это может быть и обычная книга с расширением `.xls`. Следующий код создает новую, не сохраненную книгу, которая называется SalesDataX, где X — это последовательное число, увеличивающееся при создании книг на основе этого шаблона так же, как Excel создает книги, которые называются Книга1, Книга2 и т.д. при создании книги через пользовательский интерфейс.

```
Set Workbook1 = Workbooks.Add(Template:="C:\Data\SalesData.xls")
```

Для добавления существующего файла книги в коллекцию Workbooks можно воспользоваться методом Open. Возвращаемое значение метода Open можно применить для создания объектной переменной и обращения к книге в дальнейшем.

```
Set Workbook1 = Workbooks.Open(Filename:="C:\Data\SalesData1.xls")
```

Получение имени файла из полного пути

При работе с книгами средствами VBA часто приходится указывать путь и имя файла. Некоторые задачи требуют указания только пути, например, при указании принятого по умолчанию каталога. Некоторые задачи требуют только имени файла, например, при активизации открытой книги. Для выполнения отдельных задач требуется и то и другое, например при открытии существующего файла книги, который находится за пределами текущего каталога.

После открытия книги можно получить путь, полный путь и имя файла или только имя файла. Например, следующий код отображает имя файла `SalesData1.xls` в окне сообщения:

```
Set Workbook1 = Workbooks.Open(FileName:="C:\Data\SalesData1.xls")
MsgBox Workbook1.Name
```

Свойство `Workbook1.Path` возвращает значение `"C:\Data"`, а свойство `Workbook1.FullName` возвращает значение `"C:\Data\SalesData1.xls"`.

Но если при наличии полного пути к книге необходимо определить, открыта ли книга в данный момент, из полного пути придется извлечь имя файла для получения значения свойства Name объекта Workbook. Следующая функция GetFileName возвращает имя файла `SalesData1.xls`, извлеченное из полного пути `C:\Data\SalesData1.xls`:

```
Function GetFileName(ByVal Path As String) As String
    Dim I As Integer

    For I = Len(Path) To 1 Step -1
        If Mid(Path, I, 1) = Application.PathSeparator Or _
        Mid(Path, I, 1) = ":" Then Exit For
    Next I

    GetFileName = Right(Path, Len(Path) - I)
End Function
```

Для правильной работы функции GetFileName на платформах Macintosh и Windows символ-разделитель компонентов пути извлекается из свойства PathSeparator объекта Application. (Явная проверка наличия двоеточия добавлена для обработки путей вида c:autoexec.bat, которые вполне допускаются в операционной системе Windows.) На платформе Macintosh это свойство возвращает значение :, а на платформе Windows — значение \. Функция Len возвращает количество символов в значении свойства Path, а цикл For... Next выполняет обратный перебор от последнего символа в значении Path, пока не найдет символ-разделитель. После обнаружения символа-разделителя цикл For... Next завершает свою работу. Значение счетчика I соответствует положению символа-разделителя. Если последний не обнаружен, по завершении цикла For... Next счетчик будет иметь значение 0.

> При нормальном завершении цикла For... Next переменная счетчика не будет равна значению параметра цикла Stop. Значение счетчика будет на единицу больше.

Функция Right используется для извлечения символов справа от разделителя в строке Path. Если разделитель не обнаружен, возвращаются все символы из строки Path. После получения имени файла книги можно воспользоваться функцией IsWorkbookOpen и определить, присутствует ли интересующая книга в коллекции Workbooks.

```
Public Function IsWorkbookOpen(ByVal WorkbookName As String)
↳As Boolean
    On Error Resume Next
    IsWorkbookOpen = Workbooks(WorkbookName) Is Nothing = False
End Function
```

В показанном выше коде функция IsWorkbookOpen пытается присвоить объектной переменной ссылку на книгу. После этого проверяется успешность попытки. Альтернативным способом получения этого результата является поиск в коллекции Workbooks имени объекта Workbook, которое передается в качестве аргумента.

В предыдущем примере оператор On Error Resume Next защищает от ошибок времени выполнения, если книга не открывается. Если интересующий документ найден, функция IsWorkbookOpen возвращает значение True. Если не определить возвращаемое значение функции типа Boolean, функция будет возвращать значение False. Другими словами, если книга с интересующим именем не найдена, возвращается значение False.

В следующем коде используются показанные выше определенные пользователем функции GetFileName и IsWorkbookOpen. Подпрограмма ActivateWorkbook активизирует файл книги, путь к которому присвоен переменной FileName.

```
Public Sub ActivateWorkbook()
    Dim FullName As String
    Dim FileName As String
    Dim Workbook1 As Workbook

    FullName = "C:\Temp\Book1.xls"
    FileName = GetFileName("Book1.xls")

    If IsWorkbookOpen(FileName) Then
        Set Workbook1 = Workbooks(FileName)
        Workbook1.Activate
     Else
        Set Workbook1 = Workbooks.Open(FileName:=FullName)
     End If
End Sub
```

Для извлечения имени файла книги в подпрограмме ActivateWorkbook используется функция GetFileName. Полное имя хранится в переменной FullName. Имя книги присваивается переменной FileName. После этого функция IsWorkbookOpen проверяет, открыта ли книга Book1.xls в данный момент. Если файл открыт, ссылка на объект Workbook присваивается объектной переменной Workbook1 и книга активизируется. Если файл не открыт, подпрограмма открывает файл и присваивает возвращаемое значение метода Open объектной переменной Open. После открытия книги она автоматически становится активной.

Обратите внимание, что в показанном выше коде делается предположение, что файл книги существует в указанном каталоге. Если файл отсутствует, работа кода завершится ошибкой. В разделе "Перезапись существующей книги" приводится пример функции FileExists, которую можно использовать для проверки существования файла.

Файлы в том же каталоге

Распространенным подходом является разделение приложения на несколько книг и хранение связанных книг в одном и том же каталоге. В этом же каталоге хранится книга с кодом управления приложением. В таком случае имя общего каталога можно использовать в коде для открытия связанных книг. Но если жестко вписать имя каталога в код, то при изменении имени каталога или копировании файлов в другой каталог на том же компьютере или на другом могут возникнуть проблемы. В таком случае придется модифицировать путь к каталогу в коде макросов.

Для избежания проблем с поддержкой можно воспользоваться ссылкой ThisWorkbook.Path. ThisWorkbook содержит ссылку на книгу, содержащую код. Вне зависимости от того, где расположена книга, свойство Path объекта ThisWorkbook возвращает путь к каталогу со связанными файлами. Использование этого свойства показано в следующем коде:

```
Public Sub ActivateWorkbook2()
    Dim Path As String
    Dim FileName As String
    Dim FullName As String
    Dim Workbook1 As Workbook

    FileName = "Book1.xls"

    If IsWorkbookOpen(FileName) Then
        Set Workbook1 = Workbooks(FileName)
        Workbook1.Activate
    Else
        Path = ThisWorkbook.Path
        FullName = Path & "\" & FileName
        Set Workbook1 = Workbooks.Open(FileName:=FullName)
    End If
End Sub
```

Перезапись существующей книги

При сохранении книги с помощью метода SaveAs с использованием конкретного имени файла есть вероятность, что имя уже существует на диске. Если файл существует, пользователь получает сообщение с предупреждением и должен принять решение о перезаписи существующего файла. При необходимости можно отключить предупреждение и установить программный контроль над процессом сохранения.

Если существующий файл необходимо перезаписывать при каждом сохранении, отключите предупреждение с помощью следующего кода:

```
Public Sub SaveAsTest()
    Dim Workbook1 As Workbook
    Set Workbook1 = Workbooks.Add
    Application.DisplayAlerts = False
    Workbook1.SaveAs FileName:=ThisWorkbook.Path & "\temp.xls"
    Application.DisplayAlerts = True
End Sub
```

Если необходимо проверить существование файла и выполнить альтернативную операцию, можно воспользоваться функцией Dir. Если эта проверка должна выполняться достаточно часто, можно применять следующую функцию FileExists:

```
Function FileExists(ByVal FileName As String) As Boolean
    FileExists = Len(Dir(FileName)) > 0
End Function
```

Функция Dir сравнивает аргумент со списком существующих файлов. На платформе Windows функция Dir может использоваться с символами шаблонов, например "*.xls". Если функция обнаруживает соответствующие шаблону файлы, возвращается первое совпадение. Вызов функции без аргументов позволяет получить последующие совпадения. В данном случае необходимо получить точное совпадение. Если такой файл существует, функция вернет значение, совпадающее со значением аргумента. Если такого файла нет, функция вернет строку нулевой длины. В следующем примере показано, как использовать функцию FileExists для поиска конкретного имени файла и выполнения альтернативной последовательности действий.

```
Public Sub TestForFile()

    Dim FileName As String
    FileName = ThisWorkbook.Path & "\temp.xls"

    If FileExists(FileName) Then
        MsgBox FileName & " существует"
    Else
        MsgBox FileName & " не существует"
    End If

End Sub
```

Альтернативная последовательность действий зависит от конкретной ситуации. Одним из вариантов является запрос у пользователя другого имени файла. Еще одним решением является создание нового имени файла с помощью добавления последовательности цифр к существующему имени. Пример такого решения показан ниже:

```
Public Sub CreateNextFileName()
    Dim Workbook1 As Workbook
    Dim I As Integer
    Dim FileName As String

    Set Workbook1 = Workbooks.Add(Template:=ThisWorkbook.Path &
↳"\Temp.xls")
    I = 0
    Do
        I = I + 1
        FileName = ThisWorkbook.Path & "\Temp" & I & ".xls"
    Loop While FileExists(FileName)
```

```
    Workbook1.SaveAs FileName:=FileName
End Sub
```

В этом примере при существовании сгенерированного имени файла на каждой итерации цикла `Do... Loop` значение счетчика `I` увеличивается на единицу. Когда счетчик `I` достигнет значения, не соответствующего существующему файлу, выполнение цикла завершится и файл сохранится с новым именем.

Сохранение изменений

Книгу можно закрыть с помощью метода `Close` объекта `Workbook`.

```
ActiveWorkbook.Close
```

Если в книгу вносились изменения, пользователь получит запрос на их сохранение до закрытия книги. Существует несколько способов отключения этого запроса. Их применение зависит от необходимости сохранять внесенные изменения.

Один из параметров метода `Close` позволяет автоматически сохранять изменения.

```
Sub CloseWorkbook()
    Dim Workbook1 As Workbook

    Set Workbook1 = Workbooks.Open(FileName:=ThisWorkbook.Path &
↳"\Temp.xls")
    Range("A1").Value = Format(Date, "ddd mmm dd, yyyy")
    Range("A1").EntireColumn.AutoFit
    Workbook1.Close SaveChanges:=True

End Sub
```

Если изменения не должны сохраняться, установите аргумент `SaveChanges` метода `Close` в значение `False`.

Иногда возникает необходимость оставить измененную книгу открытой для просмотра, не сохранять внесенные изменения и не получать запросов на сохранение при закрытии книги. Для этого можно установить свойство `Saved` объекта книги в значение `True`. В результате Excel посчитает, что книга не содержит изменений, которые необходимо сохранить. Перед использованием следующей строки кода необходимо удостовериться, что это действительно то, что нужно:

```
ActiveWorkbook.Saved = True
```

Коллекция Sheets

Внутри объекта `Workbook` хранится коллекция `Sheets`, в которую могут входить как объекты `Worksheet`, так и объекты `Chart`. Для совместимости со старыми версиями Excel коллекция поддерживает хранение таких объектов, как `DialogSheets`, `Excel4MacroSheets` и `Excel4InternationalMacroSheets`. В Excel 5 и Excel 95 модули также хранились в коллекции `Sheets`, но с выходом Excel 97 они переместились в редактор VBE.

> Модули из книг, созданных в Excel 5 или Excel 95, в более поздних версиях Excel считаются членами скрытой коллекции `Modules`. Для управления модулями можно продолжать использование кода, созданного для более ранних версий.

Объекты Worksheet и объекты Chart принадлежат собственным коллекциям Worksheets и Charts соответственно. В коллекцию Charts входят только листы диаграмм. Другими словами, встроенные в листы диаграммы не являются членами коллекции Charts. Встроенные в листы диаграммы хранятся в объектах ChartObject, которые входят в коллекцию ChartObjects. Дополнительная информация приводится в главе 24.

Листы

К листу можно обращаться по имени или по порядковому номеру в коллекциях Sheets или Worksheets. Если известно имя листа, то его можно использовать для выбора листа из коллекции Worksheets. Если необходимо обработать все элементы коллекции Worksheets, например в цикле For... Next, на каждый лист можно ссылаться по порядковому номеру.

Порядковый номер листа в коллекции Worksheets может отличаться от порядкового номера листа в коллекции Sheets. В пределах книги на лист Sheet1 можно ссылаться с помощью следующих конструкций:

```
ActiveWorkbook.Sheets("Sheet1")
ActiveWorkbook.Worksheets("Sheet1")
ActiveWorkbook.Sheets(2)
ActiveWorkbook.Worksheets(1)
```

Свойство Index объекта Worksheet скрывает в себе ловушку. Свойство Index объекта Worksheet содержит номер листа в коллекции Sheets, а не в коллекции Worksheets. Следующий код перебирает все листы в коллекции и выводит имя и номер листа (рис. 18.1).

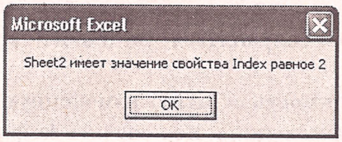


Рис. 18.1. Имя и номер листа в коллекции

```
Public Sub WorksheetIndex()
    Dim I As Integer

    For I = 1 To ThisWorkbook.Worksheets.Count
        MsgBox ThisWorkbook.Worksheets(I).Name & _
        " имеет значение свойства Index равное " &
↳ThisWorkbook.Worksheets(I).Index
    Next I
End Sub
```

По возможности избегайте использования свойства Index объекта Worksheet, так как в результате получается сложный для понимания код. В следующем примере показано, как нужно использовать значение свойства Index объекта Worksheet в качестве номера в коллекции Sheets (а не в коллекции Worksheets). Эта подпрограмма добавляет пустой лист диаграммы слева от каждого листа в активной книге.

```
Public Sub InsertChartsBeforeWorksheets()
    Dim Worksheet1 As Worksheet
```

```
    For Each Worksheet1 In Worksheets
        Charts.Add Before:=Sheets(Worksheet1.Index)
    Next Worksheet1
End Sub
```

В большинстве случаев применения свойства Index можно избежать. Показанный выше код можно переписать следующим образом:

```
Public Sub InsertChartsBeforeWorksheets2()
    Dim Worksheet As Worksheet

    For Each Worksheet In Worksheets
        Charts.Add Before:=Worksheet
    Next Worksheet1
End Sub
```

Странно, но Excel не позволяет добавлять новую диаграмму после последнего листа, хотя перемещение существующей диаграммы после последнего листа поддерживается. Если необходимо добавить лист диаграммы после каждого листа, можно воспользоваться следующим кодом:

```
Public Sub InsertChartsAfterWorksheets()
    Dim Worksheet1 As Worksheet
    Dim Chart1 As Chart
    For Each Worksheet1 In Worksheets
        Set Chart1 = Charts.Add
        Chart1.Move After:=Worksheet1
    Next Worksheet1
End Sub
```

Более подробно листы диаграмм рассматриваются в главе 23.

Методы Copy и Move

Эти методы объекта Worksheet позволяют копировать или перемещать один или несколько листов в результате одной операции. Оба метода принимают два необязательных параметра, описывающих пункт назначения операций. Пункт назначения может находиться до или после указанного листа. Если не указывать ни один из этих параметров, лист будет скопирован или перемещен в новую книгу.

Методы Copy и Move не возвращают значение или ссылку, поэтому для создания объектной переменной, которая ссылается на перемещенный или скопированный лист, придется воспользоваться другими методиками. Обычно это не проблема, так как первый лист, полученный в результате работы метода Copy, или первый лист, полученный в результате перемещения группы листов, становится активным сразу после завершения операции.

Предположим, есть книга (рис. 18.2), в которую необходимо добавить еще один лист для февраля (и еще листы для последующих месяцев). Числа в строках 3 и 4 являются входными данными. В строке 5 содержатся расчеты разности значений из строк 3 и 4. При копировании листа может потребоваться удаление данных с сохранением заголовков и формул.

В результате работы следующего кода создается новый лист с ежемесячными данными. Новый лист вставляется в книгу после последнего месяца. Код копирует первый лист, удаляет числовые данные, сохраняя заголовки и формулы. После этого лист переименовывается в соответствии с текущим месяцем и годом.

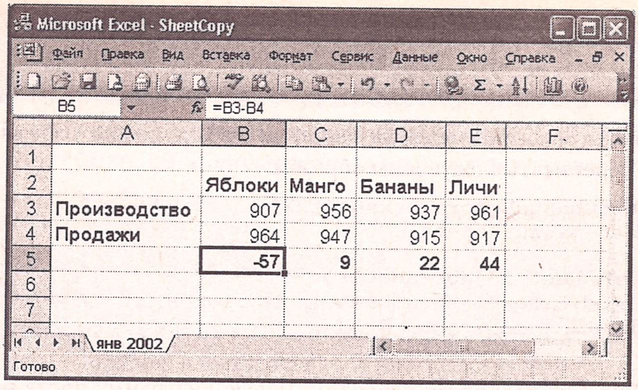


Рис. 18.2. Книга, в которую необходимо добавить лист

```
Public Sub NewMonth()
    Dim Worksheet1 As Worksheet
    Dim FirstDate As Date
    Dim FirstMonth As Integer
    Dim FirstYear As Integer

  ' Скопировать первый лист и получить ссылку
  ' на новый лист
    Worksheets(1).Copy After:=Worksheets(Worksheets.Count)
    Set Worksheet1 = Worksheets(Worksheets.Count)

  ' Считать данные с листа
    FirstDate = DateValue(Worksheets(1).Name)
    FirstMonth = Month(FirstDate)
    FirstYear = Year(FirstDate)

  ' Вычислить следующую дату и сформировать
  ' строку для имени нового листа
    Worksheet1.Name = Format(DateSerial(FirstYear, _
        FirstMonth + Worksheets.Count - 1, 1), "mmm yyyy")

  ' Удалить старое содержимое из нового листа
    On Error Resume Next
    Worksheet1.Cells.SpecialCells( _
        xlCellTypeConstants, 1).ClearContents
End Sub
```

Результат работы кода показан на рис. 18.3.

Метод `NewMonth` создает новый лист и устанавливает на него ссылку. После этого считывается и разбирается имя первого листа, на основе которого создается имя следующего листа (дата в имени листа увеличивается на один месяц). И наконец, в новом листе удаляются старые данные с сохранением заголовков и формул.

Метод `DateSerial` используется для расчета следующей даты, а метод `Format` позволяет получить строку даты в формате `"ммм гггг"`. В результате дата будет состоять из трех символов названия месяца и четырех цифр года. И наконец, с помощью метода `SpecialCells` очищаются ячейки, содержащие цифровые данные. Метод `SpecialCells` более подробно рассматривается в главе 19.

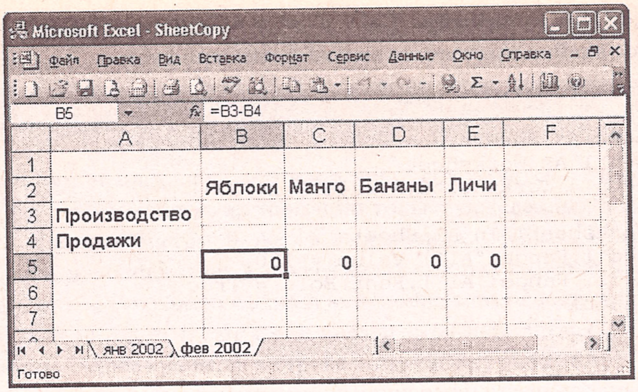


Рис. 18.3. Оригинальный лист и его копия

Группирование листов

Для ручного группирования листов в книге щелкните на вкладке листа и, удерживая клавиши <Shift> или <Ctrl>, щелкните на другом листе. Клавиша <Shift> позволяет сгруппировать все листы между выбранными. Клавиша <Ctrl> позволяет добавить в группу только выбранные листы. Кроме этого, листы можно группировать средствами кода VBA. Для этого применяется метод `Select` коллекции `Worksheets` в комбинации с функцией `Array`. Следующий код позволяет сгруппировать первый и второй листы. В результате работы кода первый лист становится активным:

```
Worksheets(Array(1, 2)).Select
Worksheets(1).Activate
```

Кроме этого, для создания группы можно воспользоваться методом `Select` объекта `Worksheet`. Первый лист выделяется обычным способом. Остальные листы добавляются в группу вызовом метода `Select` с параметром `Replace`, установленным в значение `False`:

```
Public Sub Groupsheets()
    Dim Names(1 To 3) As String
    Dim I As Integer

    Names(1) = "Янв 2002"
    Names(2) = "Фев 2002"
    Names(3) = "Мар 2002"
    Worksheets(Names(1)).Select

    For I = 2 To 3
        Worksheets(Names(I)).Select Replace:=False
    Next I
End Sub
```

Показанная выше методика особенно полезна, если имена листов предоставлены пользователем, например, через список с поддержкой множественного выделения.

> Одним из преимуществ ручного группирования листов является распространение вставки данных и изменений форматирования активного листа на остальные листы в группе. При группировании листов средствами VBA внесенные изменения сохраняются только на активном листе. Если необходимо внести одинаковые изменения на все листы в группе, воспользуйтесь циклом `For... Next`.

Следующий код вносит значение 100 в ячейку A1 листов с номерами 1, 2 и 3. Кроме этого, содержимое ячейки выделяется полужирным шрифтом.

```
Public Sub FormatGroup()

    Dim AllSheets As Sheets
    Dim Worksheet1 As Worksheet
    Set AllSheets = Worksheets(Array(1, 2, 3))

    For Each Worksheet1 In AllSheets
        Worksheet1.Range("A1").Value = 100
        Worksheet1.Range("A1").Font.Bold = True
    Next Worksheet1

End Sub
```

Объект Window

Для получения списка сгруппированных листов можно воспользоваться свойством `SelectedSheets` объекта `Window`. Можно подумать, что свойство `SelectedSheets` должно быть свойством объекта `Workbook`, но это не так. Это связано с возможностью открытия одной книги в нескольких окнах (рис. 18.4).

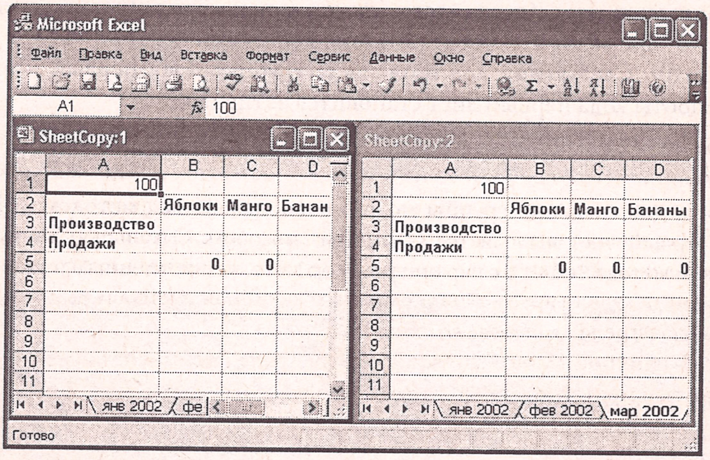


Рис. 18.4. Одна книга, открытая в нескольких окнах

Существует множество общих свойств книг и листов, которые на самом деле являются свойствами объекта `Window`. Например, `ActiveCell`, `DisplayFormulas`, `DisplayGridlines`, `DisplayHeadings` и `Selection` являются свойствами объекта `Window`. Полный список свойств этого объекта можно найти в приложении А.

Следующий код определяет выделенные ячейки активного листа, выделяет их красным цветом и устанавливает красный цвет шрифта в соответствующих диапазонах других листов группы.

```
Public Sub FormatSelectedGroup()
    Dim Sheet As Object
    Dim Address As String

    Address = Selection.Address
```

```
    For Each Sheet In ActiveWindow.SelectedSheets
        If TypeName(Sheet) = "Worksheet" Then
            Sheet.Range(Address).Font.Color = vbRed
        End If
    Next Sheet
End Sub
```

Адрес выделенного диапазона на активном листе сохраняется в строковой переменной Address. Существует возможность активизации только выделенных листов и применения форматирования красным цветом только в выделенных ячейках. Режим группирования обеспечивает выделение одних и тех же ячеек на всех листах в группе. Но активизация листа — медленный процесс. Сохранение адреса выделения в строковой переменной позволяет формировать ссылки на один и тот же диапазон в разных листах с помощью свойства Range других листов. Адрес хранится в строке вида "$B$2:$E$2,$A$3:$A$4" и не обязательно должен соответствовать единственному непрерывному диапазону.

Подпрограмма FormatSelectedGroup поддерживает включение в группу листов диаграмм и листов других типов. Перед применением форматирования подпрограмма проверяет значение свойства TypeName, которое должно быть равно "Worksheet".

> Если объектная переменная Sheet должна ссылаться на листы различных типов, ее необходимо объявить, как имеющую универсальный тип Object. В объектной модели Excel существует коллекция Sheets, но не существует объекта Sheet.

Синхронизация листов

При перемещении от одного листа к другому активизируемый лист имеет конфигурацию, которую он имел в момент последнего использования. Верхняя левая ячейка, выделенный диапазон и активная ячейка будут такими же, какими они были во время последнего обращения к листу (если лист не входит в группу). Если лист входит в группу, выделение и активная ячейка синхронизируются на всех листах в пределах группы. Но положение верхней левой ячейки не синхронизируется даже в пределах группы, поэтому при активизации листа существует вероятность, что активная ячейка и выделенный диапазон могут оказаться за пределами области видимости.

Для полной синхронизации листов даже за пределами группы следующий код необходимо добавить в модуль книги ThisWorkbook.

```
Option Explicit

Dim OldSheet As Object

Private Sub Workbook_SheetDeactivate(ByVal Sht As Object)
 'Если деактивизирован лист
 'сохранить ссылку на лист в переменной OldSheet
    If TypeName(Sht) = "Worksheet" Then Set OldSheet = Sht
End Sub

Private Sub Workbook_SheetActivate(ByVal NewSheet As Object)
    Dim CurrentColumn As Long
    Dim CurrentRow As Long
    Dim CurrentCell As String
    Dim CurrentSelection As String

    On Error GoTo Finally
    If OldSheet Is Nothing Then Exit Sub
```

```
    If TypeName(NewSheet) <> "Worksheet" Then Exit Sub
    Application.ScreenUpdating = False
    Application.EnableEvents = False

    OldSheet.Activate ' Получить старую конфигурацию листа
    CurrentColumn = ActiveWindow.ScrollColumn
    CurrentRow = ActiveWindow.ScrollRow
    CurrentSelection = Selection.Address
    CurrentCell = ActiveCell.Address

    NewSheet.Activate ' Установить новую конфигурацию листа
    ActiveaWindow.ScrollColumn = CurrentColumn
    ActiveWindow.ScrollRow = CurrentRow
    Range(CurrentSelection).Select
    Range(CurrentCell).Activate
Finally:
    Application.EnableEvents = True
End Sub
```

Оператор `Dim OldSheet as Object` должен находиться в начале модуля в области объявлений. При этом `OldSheet` будет переменной уровня модуля и будет сохранять значение, пока книга открыта. Кроме этого, доступ к переменной получат две процедуры обработки событий. Процедура обработки события `Workbook_SheetDeactivate` используется для сохранения ссылки на любой деактивизирующийся лист. Событие деактивизации возникает после активизации другого листа, поэтому сохранять свойства активного листа поздно. Параметр процедуры `Sht` ссылается на деактивизированный лист. Значение этого параметра присваивается переменной `OldSheet`.

Процедура обработки события `Workbook_SheetActivate` вызывается после процедуры обработки события `Deactivate`. Оператор `On Error GoTo Finally` обеспечивает переход на метку `Finally:` в случае ошибки без выдачи сообщений об ошибке. После метки включается обработка событий (на всякий случай, если обработка событий была отключена).

Первый оператор `If` проверяет существование определения переменной `OldSheet`. Если такое определение существует, лист был деактивизирован в течение текущего сеанса. Второй оператор `If` проверяет, является ли активизированный лист листом электронной таблицы. Если любое из условий не выполняется, процедура завершает свою работу. Эти проверки обеспечивают возможность активизации и деактивизации листов других типов.

После этого для минимизации мерцания отключается обновление экрана. Полностью отключить мерцание невозможно, так как новый лист уже активизирован и пользователь заметит фрагмент конфигурации перед изменением. После этого для предотвращения цепной реакции отключается обработка событий. Для получения необходимых данных процедура должна реактивизировать деактивизированный лист, поэтому в обычных условиях это привело бы к повторному запуску двух процедур обработки событий.

После реактивизации старого листа сохраняются свойства `ScrollRow` (строка в верхней части экрана) и `ScrollColumn` (столбец в левой части экрана), а также адреса текущего выделения и активная ячейка. Далее реактивизируется новый лист и его конфигурация устанавливается в соответствии с конфигурацией старого листа. Так как перед меткой `Finally:` отсутствует оператор `Exit Sub`, последний оператор всегда включает обработку событий.

Резюме

В этой главе рассматривалось множество приемов обработки книг и листов средствами кода VBA. Было показано, как:

- создавать новые книги и открывать существующие книги;
- обрабатывать сохранение файлов книг и перезапись существующих файлов;
- перемещать и копировать листы, а также взаимодействовать с режимом группирования.

Кроме этого, было показано, что доступ к некоторым возможностям книг и листов осуществляется через объект Window. Также был рассмотрен метод синхронизации листов через процедуры обработки событий книг. Более подробно эта тема рассматривалась в главе 5.

В этой главе были продемонстрированы несколько вспомогательных макросов, позволяющих проверять, открыта ли книга, извлекать имя файла из полного пути к книге, а также проверять существование файла книги.

19

Использование диапазонов

Вероятно, чаще всего в коде VBA используется объект Range. Объект Range применяется для представления единственной ячейки, прямоугольного блока ячеек или объединения нескольких прямоугольных блоков (несмежных диапазонов). Объект Range хранится внутри объекта Worksheet.

Объектная модель Excel не поддерживает трехмерные объекты Range, существующие на нескольких листах. Все ячейки из конкретного объекта Range должны находиться на одном и том же листе. Если возникла необходимость в работе с трехмерными диапазонами, объекты Range в пределах каждого листа придется обрабатывать отдельно.

В этой главе рассматриваются наиболее полезные свойства и методы объектаRange.

Методы Activate и Select

Эти методы могут вызвать определенную путаницу, так как иногда складывается впечатление, что между ними нет разницы. Для понимания того, что разница все же существует, сначала придется разобраться в различиях между свойствами ActiveCell и Selection объекта Application (рис. 19.1).

Свойство Selection ссылается на диапазон B3:E10. Свойство ActiveCell ссылается на ячейку C5. В эту ячейку будут вставляться вводимые пользователем данные. Свойство ActiveCell всегда ссылается на одну ячейку. Свойство Selection может ссылаться на единственную ячейку или на диапазон ячеек. Активная ячейка обычно находится в верхнем левом углу выделения, но может быть любой ячейкой в пределах выделенного диапазона (см. рис. 19.1). Положение активной ячейки в выделенном диапазоне можно изменить вручную с помощью клавиш <Tab>, <Enter>, <Shift+Tab> и <Shift+Enter>.

Для получения описанной ранее комбинации выделения и активной ячейки можно воспользоваться следующим кодом:

```
Public Sub SelectAndActivate()
    Range("B3:E10").Select
    Range("C5").Activate
End Sub
```

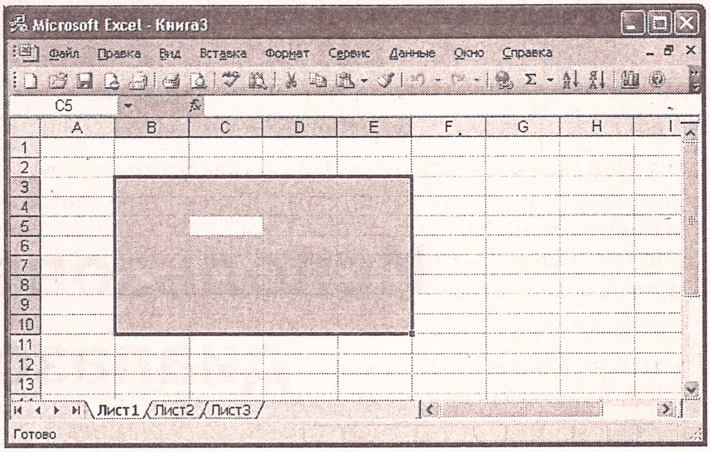


Рис. 19.1. Различие между свойствами ActiveCell и Selection

При попытке активизации ячейки за пределами выделения текущее выделение изменится и будет совпадать с активной ячейкой.

Еще одним источником неоднозначности является возможность передачи нескольких ячеек при вызове метода `Activate`. Поведение Excel определяется расположением верхней левой ячейки активизируемого диапазона. Если верхняя левая ячейка находится в пределах текущего выделения, оно не меняется и верхняя левая ячейка становится активной. Следующий код повторяет показанный ранее пример:

```
Public Sub SelectAndActivate2()
    Range("B3:E10").Select
    Range("C5:Z100").Activate
End Sub
```

Если верхняя левая ячейка активизируемого диапазона выходит за пределы текущего выделения, активизируемый диапазон становится текущим выделением, как показано в следующем примере:

```
Public Sub SelectAdActivate3()
    Range("B3:E10").Select
    Range("A2:C5").Activate
End Sub
```

В данном случае метод `Activate` имеет больший приоритет, чем метод `Select`, и диапазон A2:C5 становится текущим выделением.

> Для минимизации ошибок не рекомендуется использовать метод `Activate` для выделения диапазона ячеек. Привычка применять метод `Activate` вместо метода `Select` может привести к неожиданным результатам, если верхняя левая ячейка активизируемого диапазона находится в пределах текущего выделения.

Свойство Range

Свойство Range объекта Application можно использовать для получения ссылки на объект Range из активного листа. Следующий код возвращает ссылку на объект Range, содержащий ячейку B2 с активного листа:

```
Application.Range("B2")
```

Обратите внимание, что приведенные примеры невозможно проверить в том виде, в котором они приводятся здесь. Но если ссылаться на диапазон с активного листа, эти примеры можно тестировать в окне Immediate (Проверка) в редакторе VBE следующим образом:

```
Application.Range("B2").Select
```

Важно обратить внимание, что при отсутствии активного листа электронной таблицы использование показанной выше ссылки приведет к появлению сообщения об ошибке. Обычно такая ситуация возникает, когда активным является лист диаграммы.

Так как свойство Range объекта Application является членом коллекции <globals>, ссылку на объект Application можно опустить:

```
Range("B2")
```

Можно ссылаться на объекты Range, которые сложнее одной ячейки. В следующем примере приводится ссылка на единый блок ячеек на активном листе:

```
Range("A1:D10")
```

Этот код ссылается на несмежные диапазоны ячеек:

```
Range("A1:A10,C1:C10,E1:E10")
```

Кроме этого, свойство Range принимает два аргумента, описывающие диагонально противоположные углы диапазона. Это альтернативная возможность сослаться на диапазон A1:D10.

```
Range("A1","D10")
```

Свойство Range принимает в качестве параметров имена диапазонов. Если определить диапазон ячеек с именем SalesData, то это имя можно использовать в качестве аргумента.

```
Range("SalesData")
```

В качестве аргумента можно передавать как объекты, так и строки, что обеспечивает значительную гибкость ссылок. Например, может потребоваться создание ссылки на все ячейки столбца A от ячейки A1 до ячейки с именем LastCell.

```
Range("A1",Range("LastCell"))
```

Сокращенные ссылки на диапазоны

Для создания ссылки на диапазон ссылку вида A1 или имя диапазона можно заключить в квадратные скобки, которые являются сокращенной записью вызова метода Evaluate объекта Application. Эта запись эквивалентна использованию единственного строкового аргумента свойства Range, но этот вариант короче:

```
[B2]
[A1:D10]
[A1:A10,C1:C10,E1:E10]
[SalesData]
```

Такая форма записи часто оказывается полезной для создания абсолютных ссылок на диапазоны. Однако эта форма записи не предоставляет гибкости, характерной для свойства Range, так как не поддерживаются аргументы в виде строк и ссылок на объекты.

Диапазоны на неактивных листах

Если необходимо эффективно обрабатывать несколько листов, важно иметь возможность ссылаться на диапазоны, не активизируя листы. Переключение между листами — сложный процесс. Требующий переключения код оказывается более трудным, чем необходимо на самом деле. Переключение между листами средствами кода не имеет смысла и часто делает решения более сложными для понимания и отладки.

Все показанные ранее примеры относятся к активному листу, так как ссылки не квалифицированы ссылкой на лист. Если необходимо сослаться на диапазон за пределами активного листа, воспользуйтесь свойством Range интересующего объекта Worksheet:

```
Worksheets("Sheet1").Range("C10")
```

Если интересующий диапазон и лист находятся в неактивной книге, придется дополнительно квалифицировать ссылку на объект Range:

```
Workbooks("Sales.xls").Worksheets("Sheet1").Range("C10")
```

Если свойство Range приходится использовать в качестве аргумента другого свойства Range, необходимо соблюдать осторожность. Предположим, необходимо получить сумму значений из диапазона A1:A10 на листе Sheet1, когда активным является лист Sheet2. Может возникнуть соблазн воспользоваться следующим кодом (использование этого кода приводит к появлению ошибки времени выполнения):

```
MsgBox WorksheetFunction.Sum(Sheets("Sheet1").Range(Range("A1"), _
    Range("A10")))
```

Проблема заключается в том, что ссылки Range("A1") и Range("A10") относятся к активному листу Sheet2. Для нормальной работы кода придется использовать полностью квалифицированные ссылки:

```
MsgBox WorksheetFunction.Sum(Sheets("Sheet1").Range( _
                             Sheets("Sheet1").Range("A1"), _
                             Sheets("Sheet1").Range("A10")))
```

При создании ссылок на несколько экземпляров одного значения код можно сократить с помощью конструкции With... End With.

```
With Sheets("Sheet1")
    MsgBox WorksheetFunction.Sum(.Range(.Range("A1"), .Range("A10")))
End With
```

Свойство Range объекта Range

Обычно свойство Range используется в качестве свойства объекта Worksheet. Но можно применять свойство Range объекта Range. В таком случае свойство Range будет возвращать ссылку относительно объекта Range. Следующий код возвращает ссылку на ячейку D4:

```
Range("C3").Range("B2")
```

Если представить виртуальный лист, верхняя левая ячейка которого расположена в ячейке C3, то ячейка B2 виртуального листа будет на одну строку ниже и один столбец правее, что соответствует ячейке D4 реального листа.

Такой прием "диапазон в диапазоне" используется в коде, генерируемом при записи макроса с относительными ссылками (запись макросов рассматривалась в главе 1). Например, следующий код записан с применением относительных ссылок при выделении активной ячейки и четырех ячеек правее:

```
ActiveCell.Range("A1:E1").Select
```

Так как предыдущий код может показаться запутанным, записи с относительными ссылками стоит избегать. Свойство `Cells` является намного лучшим способом формирования относительных ссылок на ячейки.

Свойство Cells

Свойство `Cells` объектов `Application`, `Worksheet` или `Range` может использоваться для получения ссылки на объект `Range`, содержащий все ячейки из объекта `Worksheet` или объекта `Range`. Следующие строки кода ссылаются на объект `Range`, содержащий ячейки активного листа:

```
ActiveSheet.Cells
Application.Cells
```

Так как свойство `Cells` объекта `Application` входит в коллекцию `<globals>`, то на объект `Range`, содержащий все ячейки активного листа, можно ссылаться следующим образом:

```
Cells
```

Свойство `Cells` объекта `Range` можно использовать так:

```
Range("A1:D10").Cells
```

Но в этом случае свойство `Cells` возвращает ту же ссылку, что и квалифицирующий объект `Range`.

Для получения ссылки на конкретную ячейку относительно объекта `Range` можно воспользоваться свойством `Item` объекта `Range` и в качестве параметров передать относительные координаты строки и столбца. Параметр строки всегда является числовым. Параметр столбца может быть числовым или строковым (содержать букву столбца). Следующие строки кода возвращают ссылку на объект `Range`, содержащий ячейку B2 на активном листе:

```
Cells.Item(2,2)
Cells.Item(2,"B")
```

Так как свойство `Item` является принятым по умолчанию свойством объекта `Range`, обращение к свойству можно опустить:

```
Cells(2,2)
Cells(2,"B")
```

Обычно числовые параметры полезны при циклическом переборе последовательности строк или столбцов с использованием счетчика. В следующем примере выполняется перебор строк с 1 по 10 и столбцов с A по E на активном листе. В каждую ячейку заносится соответствующее значение:

```
Public Sub FillCells()
    Dim i As Integer, j As Integer

    For i = 1 To 10
        For j = 1 To 5
            Cells(i, j).Value = i * j
        Next j
    Next i
End Sub
```

Результат работы этого кода показан на рис. 19.2.

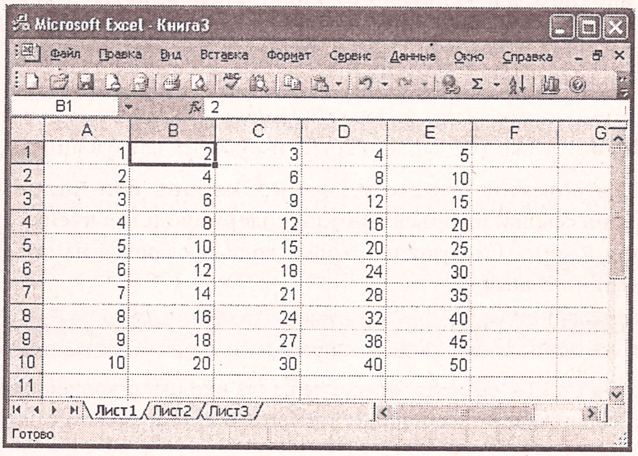


Рис. 19.2. Результат циклического перебора строк и столбцов

Использование свойства Cells в качестве параметра свойства Range

Свойство `Cells` можно использовать в качестве параметра свойства `Range` для определения объекта `Range`. Следующий код ссылается на диапазон A1:E10 активного листа:

```
Range(Cells(1,1), Cells(10,5))
```

Ссылки такого типа являются наиболее мощными, так как параметры можно указывать в виде числовых значений.

Диапазоны на неактивных листах

Как и в случае свойства `Range`, свойство `Cells` можно использовать для получения ссылки на диапазон неактивного листа:

```
Worksheets("Sheet1").Cells(2,3)
```

Применение свойства `Cells` для получения ссылки на диапазон неактивного листа требует тех же предосторожностей, что и при использовании свойства `Range`. Необходимо полностью квалифицировать свойство `Cells`. Если активным является лист `Sheet2` и интересует ссылка на диапазон A1:E10 на листе `Sheet1`, следующий код работать не будет, так как `Cells(1,1)` и `Cells(10,5)` являются свойствами активного листа:

```
Sheets("Sheet1").Range(Cells(1,1), Cells(10,5)).Font.Bold = True
```

Конструкция `With... End With` обеспечивает эффективный способ квалификации свойств:

```
With Sheets("Sheet1")
    .Range(.Cells(1, 1), .Cells(10, 5)).Font.Bold = True
End With
```

Дополнительная информация о свойстве Cells объекта Range

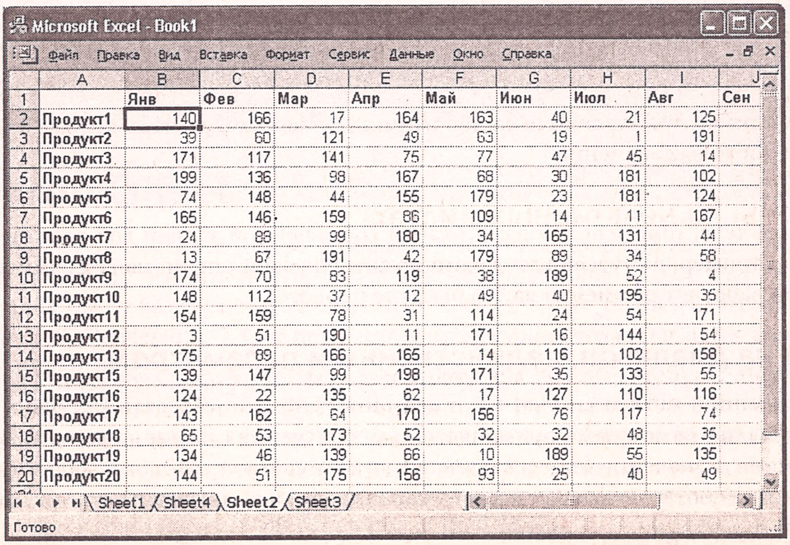
Свойство `Cells` объекта `Range` является удобным способом получения ссылки на ячейки относительно начальной ячейки или внутри блока ячеек. Следующий код ссылается на ячейку F11:

```
Range("D10:G20").Cells(2,3)
```

Если необходимо рассмотреть диапазон с именем `SalesData` и выделить красным цветом любое значение ниже 100, можно воспользоваться следующим кодом:

```
Public Sub ColorCells()
    Dim Sales As Range
    Dim I As Long
    Dim J As Long

    Set Sales = Range("SalesData")
    For I = 1 To Sales.Rows.Count
        For J = 1 To Sales.Columns.Count
            If Sales.Cells(I, J).Value < 100 Then
                Sales.Cells(I, J).Font.ColorIndex = 3
            Else
                Sales.Cells(I, J).Font.ColorIndex = 1
            End If
        Next J
    Next I
End Sub
```

Результат работы кода показан на рис. 19.3.



	A	B	C	D	E	F	G	H	I	J
1		Янв	Фев	Мар	Апр	Май	Июн	Июл	Авг	Сен
2	Продукт1	140	166	17	164	163	40	21	125	
3	Продукт2	39	60	121	49	63	19	1	191	
4	Продукт3	171	117	141	75	77	47	45	14	
5	Продукт4	199	136	98	167	68	30	181	102	
6	Продукт5	74	148	44	155	179	23	181	124	
7	Продукт6	165	146	159	86	109	14	11	167	
8	Продукт7	24	88	99	180	34	165	131	44	
9	Продукт8	13	67	191	42	179	89	34	58	
10	Продукт9	174	70	83	119	38	189	52	4	
11	Продукт10	148	112	37	12	49	40	195	35	
12	Продукт11	154	159	78	31	114	24	54	171	
13	Продукт12	3	51	190	11	171	16	144	54	
14	Продукт13	175	89	166	165	14	116	102	158	
15	Продукт15	139	147	99	198	171	35	133	55	
16	Продукт16	124	22	135	62	17	127	110	116	
17	Продукт17	143	162	64	170	156	76	117	74	
18	Продукт18	65	53	173	52	32	32	48	35	
19	Продукт19	134	46	139	66	10	189	55	135	
20	Продукт20	144	51	175	156	93	25	40	49	

Рис. 19.3. Использование свойства Cells

На самом деле интересующие ячейки не обязательно должны находиться в пределах объекта Range. Можно ссылаться на ячейки за пределами оригинального диапазона. Это значит, что достаточно знать координаты верхней левой ячейки объекта Range. Данный код ссылается на ячейку F11, как и в предыдущем примере:

```
Range("D10").Cells(2,3)
```

Кроме этого, можно воспользоваться сокращенной версией. Следующий код также ссылается на ячейку F11:

```
Range("D10")(2,3)
```

Следующий код тоже работает, так как можно обращаться непосредственно к свойству Items объекта Range, а не к свойству Cells, как было показано ранее:

```
Range("D10").Item(2,3)
```

При этом можно использовать нулевые и даже отрицательные смещения, пока ссылка не выходит за пределы листа. Иногда использование таких ссылок может привести к неожиданным результатам, например, следующий код ссылается на ячейку C9:

```
Range("D10")(0,0)
```

Данный код ссылается на ячейку B8:

```
Range("D10")(-1,-1)
```

Предыдущий пример использования свойства Font.ColorIndex диапазона Sales может быть переписан следующим образом:

```
Public Sub ColorCells()
    Dim Sales As Range
    Dim i As Long
    Dim j As Long

    Set Sales = Range("SalesData")
    For i = 1 To Sales.Rows.Count
        For j = 1 To Sales.Columns.Count
            If Sales(i, j).Value < 100 Then
                Sales(i, j).Font.ColorIndex = 4
            Else
                Sales(i, j).Font.ColorIndex = 1
            End If
        Next j
    Next i
End Sub
```

На самом деле, при использовании такой сокращенной записи код работает немного быстрее. Например, как утверждает один из авторов этой книги, на его компьютере второй пример работает примерно на 5% быстрее первого.

Ссылка на диапазон с единственным параметром

В сокращенной записи ссылки на диапазон можно использовать как один, так и два параметра. Если эта методика применяется для получения ссылки на диапазон с несколькими строками и значение индекса превышает количество столбцов в диапазоне, ссылка переносится на соответствующее количество столбцов на следующих строках диапазона.

Данный код ссылается на ячейку E10:

```
Range("D10:E11")(2)
```

Следующий код ссылается на ячейку D11:

```
Range("D10:E11")(3)
```

Значение индекса может превышать количество ячеек в объекте Range. При этом ссылка будет продолжать перемещаться по столбцам объекта Range. Следующий код ссылается на ячейку D12:

```
Range("D10:E11")(5)
```

Использование единственного параметра может потребоваться для перебора всех ячеек диапазона без обращения к отдельным строкам и столбцам. Пример подпрограммы ColorCells можно модифицировать еще больше, применив следующий прием:

```
Public Sub ColorCells()
    Dim Sales As Range
    Dim i As Long

    Set Sales = Range("SalesData")
    For i = 1 To Sales.Cells.Count
        If Sales(i).Value < 100 Then
            Sales(i).Font.ColorIndex = 5
        Else
            Sales(i).Font.ColorIndex = 1
        End If
    Next i
End Sub
```

В четвертом и последнем варианте подпрограммы ColorCells можно перебрать все ячейки диапазона с помощью цикла For Each... Next. Этот способ подходит для ситуаций, когда значение индекса цикла не используется для других операций.

```
Public Sub ColorCells()
    Dim aRange As Range

    For Each aRange In Range("SalesData")
        If aRange.Value < 100 Then
            aRange.Font.ColorIndex = 6
        Else
            aRange.Font.ColorIndex = 1
        End If
    Next aRange
End Sub
```

Свойство Offset

Свойство Offset объекта Range возвращает объекты, похожие на значение свойства Cells. Между этими объектами существует два отличия. Первое заключается в том, что значение свойства Offset отсчитывается от 0, а не от 1 (это подразумевается названием свойства "offset"/"смещение"). Обе показанные ниже строки кода ссылаются на ячейку A10:

```
Range("A10").Cells(1,1)
Range("A10").Offset(0,0)
```

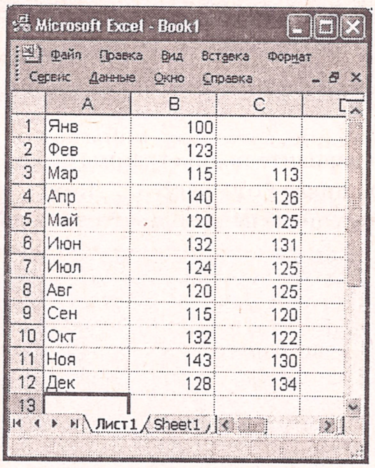
Второе отличие заключается в том, что объект Range, который возвращается в качестве значения свойства Cells, состоит из одной ячейки. Объект Range, возвращаемый свойством Offset, содержит то же количество строк и столбцов, что и исходный диапазон. Следующая строка кода ссылается на диапазон B2:C3:

```
Range("A1:B2").Offset(1,1)
```

Свойство `Offset` может потребоваться для получения ссылки на диапазон того же размера, но смещенного на определенное расстояние. Например, в диапазоне B1:B12 могут храниться суммы продаж с января по декабрь, в ячейках C3:C12 может потребоваться создание плавающего среднего значения за три месяца (с декабря по март). Эту задачу выполняет следующий код:

```
Public Sub MoveAverage()
    Dim aRange As Range
    Dim i As Long

    Set aRange = Range("B1:B3")
    For i = 3 To 12
        Cells(i, "C").Value = WorksheetFunction.Round _
            (WorksheetFunction.Sum(aRange) / 3, 0)
        Set aRange = aRange.Offset(1, 0)
    Next i
End Sub
```

Результат работы кода показан на рис. 19.4.



	A	B	C
1	Янв	100	
2	Фев	123	
3	Мар	115	113
4	Апр	140	126
5	Май	120	125
6	Июн	132	131
7	Июл	124	125
8	Авг	120	125
9	Сен	115	120
10	Окт	132	122
11	Ноя	143	130
12	Дек	128	134
13			

Рис. 19.4. Среднее значение суммы продаж за три месяца

Свойство Resize

Свойство `Resize` объекта `Range` позволяет получить ссылку на диапазон с той же верхней левой ячейкой, но другим количеством строк и столбцов. Следующий код ссылается на диапазон D10:E10:

```
Range("D10:F20").Resize(1,2)
```

Свойство `Resize` также позволяет расширить или уменьшить диапазон на строку или столбец. Например, если существует набор данных, хранящийся в диапазоне с именем `Database`, и к набору добавлена еще одна строка, имя диапазона придется переопределить для включения дополнительной строки. Следующий код увеличивает именованный диапазон на одну строку:

```
With Range("Database")
  .Resize(.Rows.Count + 1).Name = "Database"
End With
```

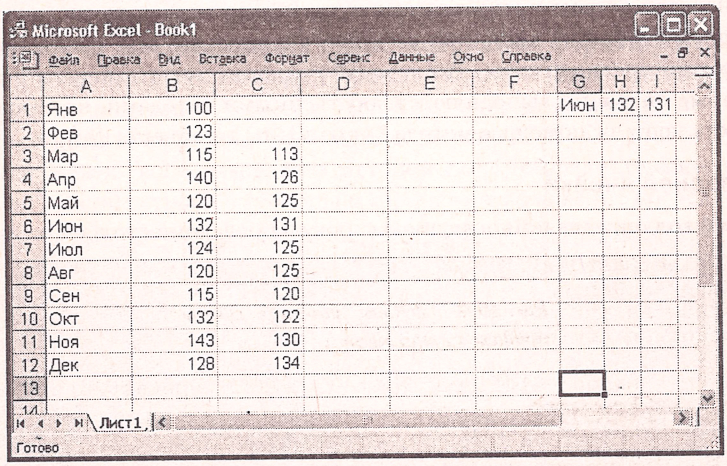
Если опустить второй параметр, количество столбцов останется неизменным. Точно так же можно опустить первый параметр и оставить неизменным количество строк. Следующий код ссылается на диапазон A1:C10:

```
Range("A1:B10").Resize(, 3)
```

Нижеприведенный код позволяет выполнять поиск значения в списке и копирование найденного значения и двух столбцов справа на новое место:

```
Public Sub FindIt()
    Dim aRange As Range

    Set aRange = Range("A1:A12").Find(What:="Июн", _
        LookAt:=xlWhole, LookIn:=xlValues)

    If aRange Is Nothing Then
        MsgBox "Данные не найдены"
        Exit Sub
    Else
        aRange.Resize(1, 3).Copy Destination:=Range("G1")
    End If
End Sub
```

Результат работы этого кода показан на рис. 19.5.



	A	B	C	D	E	F	G	H	I
1	Янв	100					Июн	132	131
2	Фев	123							
3	Мар	115	113						
4	Апр	140	126						
5	Май	120	125						
6	Июн	132	131						
7	Июл	124	125						
8	Авг	120	125						
9	Сен	115	120						
10	Окт	132	122						
11	Ноя	143	130						
12	Дек	128	134						
13									

Рис. 19.5. Поиск и копирование интересующих ячеек

Метод Find не дублирует функциональность команды меню Правка⇨Найти (Edit⇨Find). Метод возвращает ссылку на ячейку в виде объекта Range, но найденная ячейка не выделяется. Если метод Find не может обнаружить интересующую ячейку, возвращается объект null, возврат которого можно проверить с помощью оператора IsNothing. При попытке копирования объекта null возвращается ошибка времени выполнения.

Метод SpecialCells

При нажатии клавиши <F5> на открытом листе выводится диалоговое окно Переход (Go To). В диалоговом окне можно щелкнуть на кнопке Выделить (Special). После этого появится диалоговое окно, показанное на рис. 19.6.

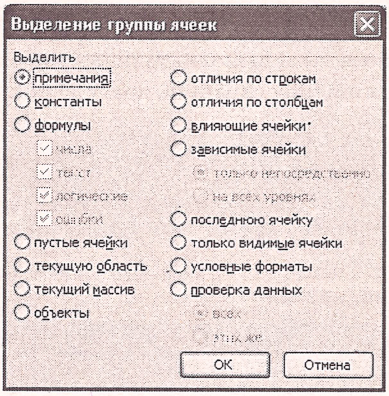


Рис. 19.6. Диалоговое окно Выделение группы ячеек

В этом диалоговом окне предоставляются несколько полезных операций, например, поиск последней ячейки на листе или всех ячеек с числами, а не вычисляемыми значениями. Несложно предположить, что все эти операции могут выполняться средствами VBA. Некоторые операции выполняются собственными методами, но большинство операций осуществляются с помощью метода `SpecialCells` объекта `Range`.

Поиск последней ячейки

Следующий код определяет последнюю строку и последний столбец на листе:

```
Public Sub SelectLastCell()
   Dim aRange As Range
   Dim lastRow As Integer
   Dim lastColumn As Integer

   Set aRange = Range("A1").SpecialCells(xlCellTypeLastCell)
   lastRow = aRange.Row
   lastColumn = aRange.Column

   MsgBox lastColumn
End Sub
```

Последняя ячейка определяется как пересечение последней содержащей информацию строки листа и последнего содержащего информацию столбца листа. В процессе поиска последней ячейки Excel просматривает и те ячейки, в которых в течение текущего сеанса хранилась информация (даже если к моменту поиска эта информация была удалена). Последняя ячейка не сбрасывается, пока лист не будет сохранен.

Считается, что содержащие форматирование и разблокированные ячейки также содержат информацию. В результате последняя ячейка может оказаться далеко за пределами диапазона, содержащего данные, особенно если книга была импортирована из другого

приложения электронных таблиц, например Lotus 1-2-3. Если же рассматриваются только те ячейки, в которых содержатся числа, текст и формулы, воспользуйтесь следующим кодом:

```
Public Sub GetRealLastCell()
    Dim realLastRow As Long
    Dim realLastColumn As Long
    Range("A1").Select

    On Error Resume Next
    realLastRow = Cells.Find("*", Range("A1"), _
    xlFormulas, , xlByRows, xlPrevious).Row
    realLastColumn = Cells.Find("*", Range("A1"), _
        xlFormulas, , xlByColumns, xlPrevious).Column
    Cells(realLastRow, realLastColumn).Select
End Sub
```

В этом примере метод `Find` выполняет обратный поиск от последней строки и столбца, содержащих любой символ до ячейки A1 (это значит, что Excel переходит на последнюю ячейку и от нее начинает поиск в сторону ячейки A1). Оператор `On Error Resume Next` позволяет предотвратить ошибки времени выполнения, если лист оказывается пустым.

> Обратите внимание, что переменные для хранения номера строки необходимо объявить с помощью оператора `Dim` как имеющие тип `Long`, а не Integer, так как переменные типа `Integer` могут хранить значения до 32767, а лист может содержать до 65536 строк.

Для избавления от дополнительных строк, содержащих форматирование, выделите целые строки и воспользуйтесь командой Правка⇨Удалить (Edit⇨Delete). Кроме этого, выделите и удалите ненужные столбцы. При этом последняя ячейка не сбрасывается. Для сброса последней ячейки лист необходимо сохранить. Кроме этого, для сброса последней ячейки можно воспользоваться методом `ActiveSheet.UsedRange`. Следующий код удаляет лишние строки и столбцы и сбрасывает положение последней ячейки:

```
Public Sub DeleteUnusedFormats()
    Dim lastRow As Long
    Dim lastColumn As Long
    Dim realLastRow As Long
    Dim realLastColumn As Long

    With Range("A1").SpecialCells(xlCellTypeLastCell)
        lastRow = .Row
        lastColumn = .Column
    End With

    realLastRow = Cells.Find("*", Range("A1"), _
         xlFormulas, , xlByRows, xlPrevious).Row
         realLastColumn = Cells.Find("*", Range("A1"), _
             xlFormulas, , xlByColumns, xlPrevious).Column

    If realLastRow < lastRow Then
        Range(Cells(realLastRow + 1, 1), _
        Cells(lastRow, 1)).EntireRow.Delete
    End If
    If realLastColumn < lastColumn Then
        Range(Cells(1, realLastColumn + 1), _
            Cells(1, lastColumn)).EntireColumn.Delete
    End If
```

```
    ActiveSheet.UsedRange
End Sub
```

Свойство EntireRow объекта Range возвращает ссылку на объект Range, который содержит все столбцы от 1 до 256 (или от A до IV) из исходного диапазона. Свойство EntireColumn возвращает ссылку на объект Range, содержащий все строки (от 1 до 65536) столбцов, входящих в исходный диапазон.

Удаление чисел

Иногда возникает потребность в удалении всех данных на листе или в шаблоне, чтобы явно указать на необходимость ввода новых значений. Следующий код удаляет все числа на листе, сохраняя формулы:

```
On Error Resume Next
Cells.SpecialCells(xlCellTypeConstants, xlNumbers).ClearContents
```

> Оператор On Error в показанном выше примере позволяет отключить сообщение об ошибке времени выполнения, если в интересующих ячейках отсутствуют числа.

Excel рассматривает даты как числа, поэтому в результате работы показанного кода даты тоже удаляются. Если даты используются в качестве заголовков и их необходимо сохранить, можно воспользоваться следующим кодом:

```
Public Sub ClearNonDateCells()
    Dim aRange As Range

    For Each aRange In Cells.SpecialCells(xlCellTypeConstants,
↳xlNumbers)
        If Not IsDate(aRange.Value) Then aRange.ClearContents
    Next aRange
End Sub
```

Свойство CurrentRegion

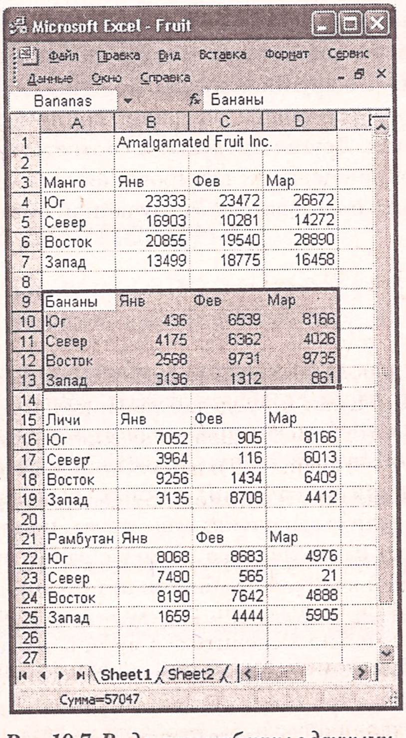
Если таблица с данными отделена от остальных данных как минимум одной пустой строкой и одним пустым столбцом, конкретную таблицу можно выделить с помощью свойства CurrentRegion любой ячейки в пределах таблицы. Это свойство является эквивалентом комбинации клавиш <Ctrl+Shift+*>. Для выделения таблицы Бананы, показанной на рис. 19.7, достаточно щелкнуть на ячейке A9 и нажать комбинацию клавиш <Ctrl+Shift+*>.

Того же результата можно добиться с помощью работы следующего кода (если ячейка A9 называется Bananas):

```
Range("Bananas").CurrentRegion.Select
```

Это свойство оказывается полезным при работе с таблицами, меняющими размер с течением времени. При этом можно выделить все месяцы до текущего с ростом таблицы в течение года, а код каждый месяц модифицировать не нужно. Обычно в процессе работы кода выделение вообще не требуется. Если необходимо выполнить консолидацию данных о фруктах в единую таблицу на листе Consolidation, а верхний левый угол каждой таблицы называется в соответствии с названием продукта, то для консолидации можно воспользоваться следующим кодом:

```
Public Sub Consolidate()
    Dim Products As Variant
    Dim Source As Range
    Dim Destination As Range
    Dim i As Long

    Application.ScreenUpdating = False
    Products = Array("Mangoes", "Bananas", "Lychees", "Rambutan")
    Set Destination = Worksheets("Consolidation").Range("B4")
    For i = LBound(Products) To UBound(Products)
        With Range(Products(i)).CurrentRegion
            Set Source = .Offset(1, 1).Resize(.Rows.Count - 1,
↳.Columns.Count - 1)
        End With
        Source.Copy
        If i = LBound(Products) Then
            Destination.PasteSpecial xlPasteValues, _
xlPasteSpecialOperationNone
        Else
            Destination.PasteSpecial xlPasteValues,
↳xlPasteSpecialOperationAdd
        End If
    Next i
    Application.CutCopyMode = False 'очистить буфер
End Sub
```



	A	B	C	D
1		Amalgamated Fruit Inc.		
2				
3	Манго	Янв	Фев	Мар
4	Юг	23333	23472	26672
5	Север	16903	10281	14272
6	Восток	20855	19540	28890
7	Запад	13499	18775	16458
8				
9	Бананы	Янв	Фев	Мар
10	Юг	436	6539	8166
11	Север	4175	6362	4026
12	Восток	2568	9731	9735
13	Запад	3136	1312	861
14				
15	Личи	Янв	Фев	Мар
16	Юг	7052	905	8166
17	Север	3964	116	6013
18	Восток	9256	1434	6409
19	Запад	3135	8708	4412
20				
21	Рамбутан	Янв	Фев	Мар
22	Юг	8068	8683	4976
23	Север	7480	565	21
24	Восток	8190	7642	4888
25	Запад	1659	4444	5905
26				
27				

Рис. 19.7. Выделение таблицы с данными

Результат работы кода показан на рис. 19.8.

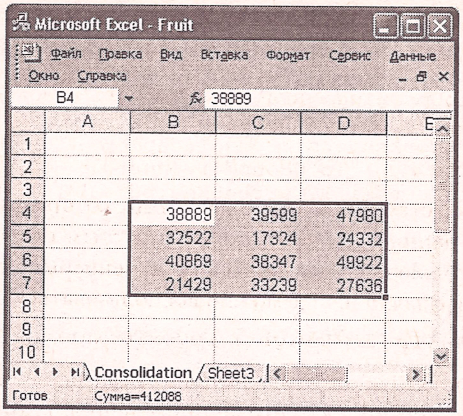


Рис. 19.8. Консолидация таблиц с данными в одну таблицу

Для ускорения работы кода и защиты от мерцания обновление экрана отключается. Функция `Array` обеспечивает удобный способ определения относительно коротких списков обрабатываемых элементов. Функции `LBound` и `UBound` используются для обеспечения независимости от оператора `Option Base` в начале модуля. Этот код беспрепятственно может использоваться в других модулях.

Первый продукт и соответствующие значения копируются поверх существующих значений в указанных ячейках. Другие продукты и соответствующие значения копируются в соседние ячейки. В конце операции копирования выполняется очистка буфера обмена, что позволяет защитить пользователя от случайной повторной вставки через нажатие клавиши <Enter>.

Свойство End

Это свойство эмулирует результат нажатия комбинации клавиш <Ctrl+стрелка>. Если выделить ячейку в верхней части столбца и нажать комбинацию клавиш <Ctrl+вниз>, выделение переместится на ячейку, которая находится перед первой пустой ячейкой. Если в столбце нет ни одной пустой ячейки, выделяется последняя ячейка с данными. Если ячейка после выделенной не содержит данных, выделение перемещается на следующую ячейку с данными, если такая ячейка существует, или в конец листа.

Следующий код ссылается на последнюю ячейку с данными в конце столбца A (если между ячейкой A1 и последней ячейкой с данными нет пустой ячейки):

```
Range("A1").End(xlDown)
```

Для перемещения в обратном направлении можно воспользоваться константами `xlUp`, `xlToLeft` и `xlToRight`.

Если в данных встречаются пропуски, но необходимо получить ссылку на последнюю ячейку в столбце A, можно начинать с нижней части листа и проводить поиск в обратном направлении. Данные не должны выходить за пределы ячейки A65536:

```
Range("A65536").End(xlUp)
```

В разделе о строках далее в этой главе будет показан метод отказа от ссылки на ячейку А65536 и генерализации кода для использования в различных версиях Excel:

Получение ссылки на диапазоны через свойство End

Для получения ссылки на диапазон ячеек от активной ячейки до конца текущего столбца можно воспользоваться следующим кодом:

```
Range(ActiveCell, ActiveCell.End(xlDown)).Select
```

Предположим, что существует таблица с данными, которая начинается в ячейке B3 и отделена от остальных данных пустой строкой и пустым столбцом. Пока таблица содержит непрерывные заголовки в верхней части и непрерывные данные в последнем столбце, то для получения ссылки на нее можно воспользоваться следующим кодом:

```
Range("B3", Range("B3").End(xlToRight).End(xlDown)).Select
```

В данном случае результат работы кода эквивалентен использованию свойства CurrentRegion, но свойство End имеет и другие применения, которые показаны в следующих примерах.

Как обычно, при работе с объектом Range из кода VBA необходимость в выделении ячеек не возникает. Следующий код выполняет копирование непрерывных заголовков из верхней части листа Sheet1 в верхнюю часть листа Sheet2.

```
With Worksheets("Sheet1").Range("A1")
    .Range(.Cells(1), .End(xlToRight)).Copy Destination:= _
        Worksheets("Sheet2").Range("A1")
End With
```

Пока существует активная книга, этот код может выполняться для любого активного листа.

Суммирование диапазона

Предположим, что функция СУММ (SUM) в активной ячейке должна складывать значения из ячеек ниже до следующей пустой ячейки. Для этого можно воспользоваться следующим кодом:

```
With ActiveCell
    Set aRange = Range(.Offset(1), .Offset(1).End(xlDown))
    .Formula = "=SUM(" & aRange.Address & ")"
End With
```

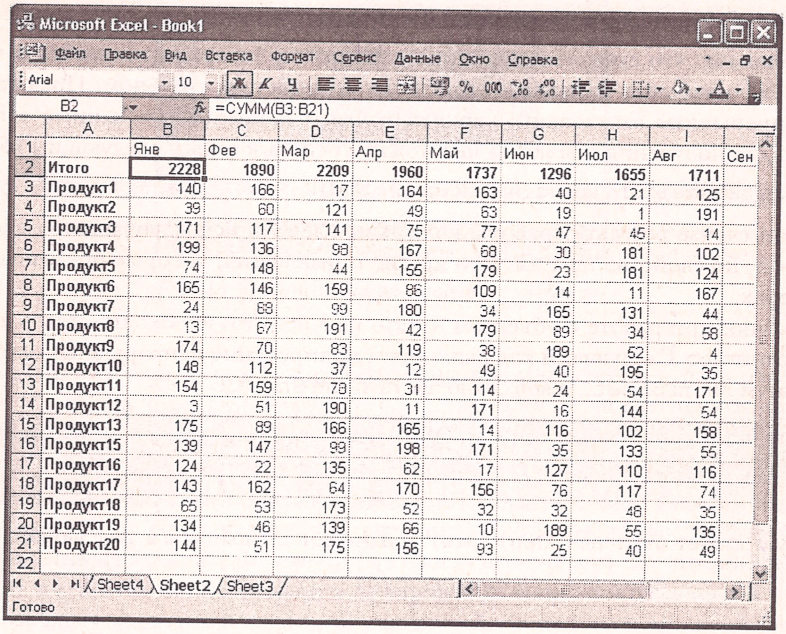
Свойство Address объекта Range по умолчанию возвращает абсолютный адрес. Если необходимо копирование формулы в другие ячейки и суммирование расположенных ниже значений, можно воспользоваться относительным адресом и выполнить копирование следующим образом:

```
Public Sub SumRangeTest()
    Dim aRange As Range

    With ActiveCell
        Set aRange = Range(.Offset(1), .Offset(1).End(xlDown))
        .Formula = "=SUM(" & aRange.Address(RowAbsolute:=False, _
            ColumnAbsolute:=False) & ")"
        .Copy Destination:=Range(.Cells(1),
```

```
↳.Offset(1).End(xlToRight).Offset(-1))
    End With

End Sub
```

Конец диапазона назначения определяется по положению последней ячейки с данными в строке с формулой СУММ (SUM).

Результат выполнения этого кода показан на рис. 19.9.



	A	B	C	D	E	F	G	H	I
1		Янв	Фев	Мар	Апр	Май	Июн	Июл	Авг
2	Итого	2228	1890	2209	1960	1737	1296	1655	1711
3	Продукт1	140	166	17	164	163	40	21	125
4	Продукт2	39	60	121	49	63	19	1	191
5	Продукт3	171	117	141	75	77	47	45	14
6	Продукт4	199	136	98	167	68	30	181	102
7	Продукт5	74	148	44	155	179	23	181	124
8	Продукт6	165	146	159	86	109	14	11	167
9	Продукт7	24	88	99	180	34	165	131	44
10	Продукт8	13	67	191	42	179	89	34	58
11	Продукт9	174	70	83	119	38	189	52	4
12	Продукт10	148	112	37	12	49	40	195	35
13	Продукт11	154	159	78	31	114	24	54	171
14	Продукт12	3	51	190	11	171	16	144	54
15	Продукт13	175	89	166	165	14	116	102	158
16	Продукт15	139	147	99	198	171	35	133	55
17	Продукт16	124	22	135	62	17	127	110	116
18	Продукт17	143	162	64	170	156	76	117	74
19	Продукт18	65	53	173	52	32	32	48	35
20	Продукт19	134	46	139	66	10	189	55	135
21	Продукт20	144	51	175	156	93	25	40	49
22									

Рис. 19.9. Суммирование значений диапазона

Свойства Columns и Rows

Эти свойства предоставляются объектами Application, Worksheet и Range. Свойства возвращают ссылку на все столбцы или строки листа, или диапазона. В любом случае свойство возвращает объект Range, но возвращаемый объект Range может иметь странные характеристики, заставляющие думать, что существует "объект Column" или "объект Row" (в Excel таких объектов не существует). Эти свойства могут использоваться для подсчета количества строк или столбцов, а также для обработки всех строк или столбцов диапазона.

В Excel 97 количество строк листа увеличено с 16384 до 65536. Для определения количества строк в активном листе можно воспользоваться свойством Count объекта, возвращаемого свойством Rows:

```
Rows.Count
```

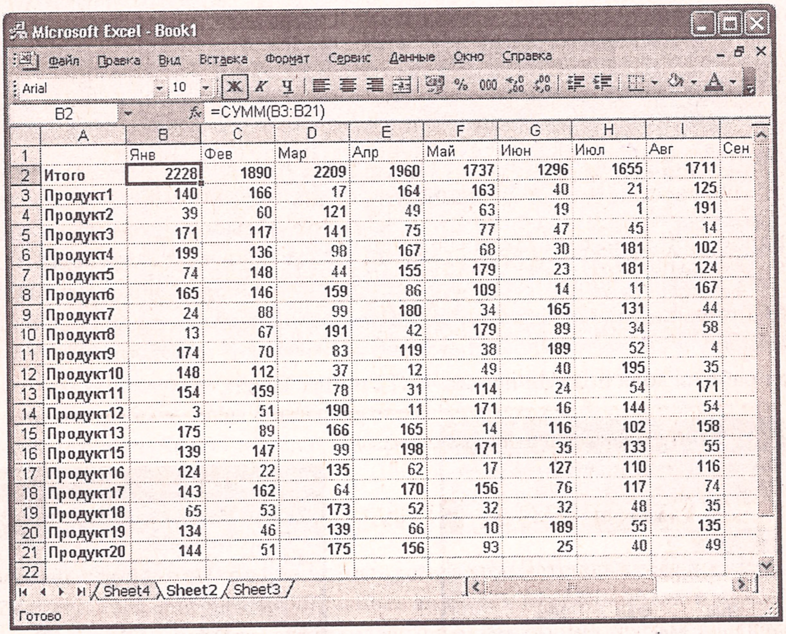
Это свойство оказывается полезным при создании макроса, работающего во всех версиях интерпретатора Excel VBA, а также для определения последней строки с данными в столбце начиная с последней строки листа:

```
Cells(Rows.Count, "A").End(xlUp).Select
```

Если в диапазоне SalesData хранится таблица с данными, следующий код позволяет перебрать все строки и выделить полужирным шрифтом те, в которых значение первой ячейки превышает 1000:

```
Public Sub BoldCells()
    Dim Row As Object
    For Each Row In Range("SalesData").Rows
        If Row.Cells(1).Value > 1000 Then
            Row.Font.Bold = True
        Else
            Row.Font.Bold = False
        End If
    Next Row

End Sub
```

Результат работы этого кода показан на рис. 19.10.



	A	B	C	D	E	F	G	H	I	
1		Янв	Фев	Мар	Апр	Май	Июн	Июл	Авг	Сен
2	Итого	2228	1890	2209	1960	1737	1296	1655	1711	
3	Продукт1	140	166	17	164	163	40	21	125	
4	Продукт2	39	60	121	49	63	19	1	191	
5	Продукт3	171	117	141	75	77	47	45	14	
6	Продукт4	199	136	98	167	68	30	181	102	
7	Продукт5	74	148	44	155	179	23	181	124	
8	Продукт6	165	146	159	86	109	14	11	167	
9	Продукт7	24	88	99	180	34	165	131	44	
10	Продукт8	13	67	191	42	179	89	34	58	
11	Продукт9	174	70	83	119	38	189	52	4	
12	Продукт10	148	112	37	12	49	40	195	35	
13	Продукт11	154	159	78	31	114	24	54	171	
14	Продукт12	3	51	190	11	171	16	144	54	
15	Продукт13	175	89	166	165	14	116	102	158	
16	Продукт15	139	147	99	198	171	35	133	55	
17	Продукт16	124	22	135	62	17	127	110	116	
18	Продукт17	143	162	64	170	156	76	117	74	
19	Продукт18	65	53	173	52	32	32	48	35	
20	Продукт19	134	46	139	66	10	189	55	135	
21	Продукт20	144	51	175	156	93	25	40	49	
22										

Рис. 19.10. Выделение строк полужирным шрифтом

Интересно, что вызов Rows.Cells(1) нельзя заменить на Row(1), как в случае с объектом Range. В данном случае использование сокращенной записи приводит к ошибке времени выполнения. Возможно, свойства Rows и Columns действительно возвращают специальный объект Range и о таких объектах проще думать как об объектах Row и Column, хотя официально в Excel таких объектов не существует.

Области

При использовании свойств Columns и Rows совместно с составными диапазонами необходимо соблюдать осторожность. (Составные диапазоны могут возвращаться методом SpecialCells при поиске ячеек с числовыми значениями или пустых ячеек на листе.) Составные диапазоны состоят из нескольких отдельных прямоугольных блоков. Ес-

ли ячейки не являются членами одного блока, вызов свойства Rows.Count возвращает количество строк только в первом блоке. Следующий код возвращает результат 5, так как рассматривается только первый диапазон, A1:B5.

```
Range("A1:B5,C6:D10,E11:F15").Rows.Count
```

Блоки составного диапазона являются объектами Range в составе коллекции Area. Эти объекты могут обрабатываться по отдельности. Ниже показан код, отображающий адрес каждого блока в объекте Range:

```
For Each aRange In Range("A1:B5,C6:D10,E11:F15").Areas
    MsgBox aRange.Address
Next Rng
```

В показанном ниже листе содержатся прогнозы объемов продаж, записанные в виде чисел. Значения стоимости рассчитываются по формулам. Следующий код копирует все числовые константы с активного листа в блоки листа Sheet3. Между каждым блоком оставляется пустая строка:

```
Public Sub CopyAreas()
    Dim aRange As Range
    Dim Destination As Range

    Set Destination = Worksheets("Sheet3").Range("A1")
    For Each aRange In Cells.SpecialCells( _
        xlCellTypeConstants, xlNumbers).Areas

        aRange.Copy Destination:=Destination
        Set Destination = Destination.Offset(aRange.Rows.Count + 1)

    Next aRange
End Sub
```

Результат работы кода показан на рис. 19.11 и рис. 19.12.

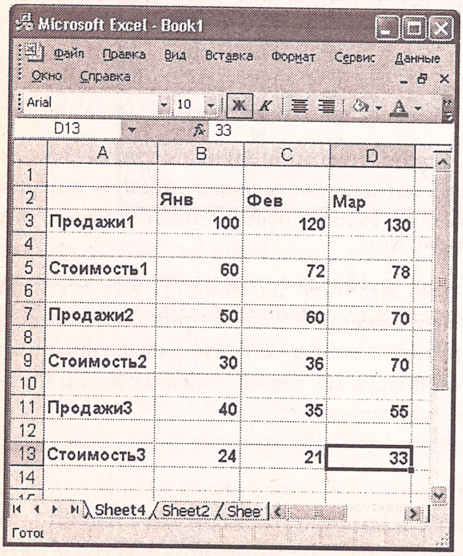


Рис. 19.11. Исходный лист с числовыми константами и формулами

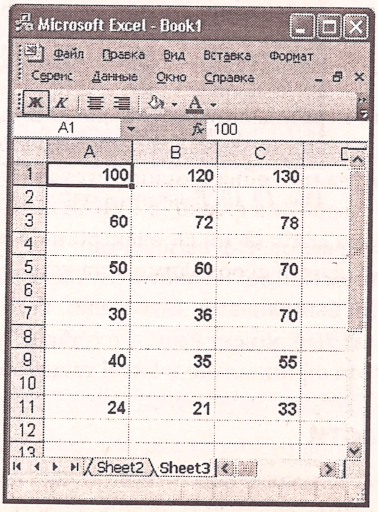


Рис. 19.12. Скопированные числовые константы

Методы Union и Intersect

Это методы объекта Application, но они могут использоваться без ссылки на этот объект, так как являются членами коллекции <globals>. Как будет показано ниже, методы Union и Intersect часто оказываются очень полезными.

Метод Union можно использовать для генерации диапазона из двух или большего количества блоков ячеек. Метод Intersect применяется для получения общих ячеек двух или большего количества диапазонов (пересечения диапазонов). Следующая процедура обработки событий из модуля кода листа не позволяет пользователю выделить ячейки из диапазонов B10:F20 и H10:L20. Одним из применений этой подпрограммы является защита данных от изменения пользователем:

```
Private Sub Worksheet_SelectionChange(ByVal Target As Range)
    Dim Forbidden As Range

    Set Forbidden = Union(Range("B10:F20"), Range("H10:L20"))
    If Intersect(Target, Forbidden) Is Nothing Then Exit Sub
    Range("A1").Select
    MsgBox "Нельзя выделить ячейки в диапазоне " & Forbidden.Address,
↳vbCritical
End Sub
```

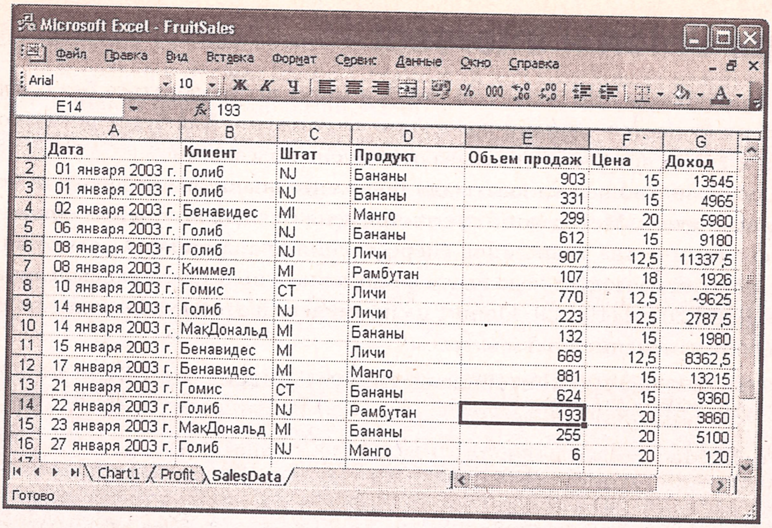
Процедуры обработки событий рассматривались в главе 2. Дополнительная информация о процедурах обработки событий приводилась в главе 10.

Процедура обработки события Worksheet_SelectionChange вызывается каждый раз, когда пользователь выделяет новый диапазон листа, связанного с модулем процедуры. В показанном выше коде метод Union используется для определения запрещенного диапазона по двум несмежным диапазонам. После этого метод Intersect и оператор If применяются для проверки пересечения выделения с запрещенным диапазоном. Если пересечение отсутствует, метод Intersect возвращает значение Nothing и подпрограмма завершает работу. Если пересечение существует, выполняется код после оператора If — выделяется ячейка A1 и выдается предупреждение.

Пустые ячейки

Для перемещения по строке или столбцу до первой пустой ячейки можно воспользоваться свойством End. Еще одним способом является перебор ячеек в цикле и завершение цикла при обнаружении пустой ячейки. Для обнаружения пустых ячеек также можно воспользоваться функцией VBA IsEmpty.

В показанной ниже электронной таблице (рис. 19.13) для получения удобного для чтения отчета необходимо вставить пустые строки между каждой неделей.



	A	B	C	D	E	F	G
1	Дата	Клиент	Штат	Продукт	Объем продаж	Цена	Доход
2	01 января 2003 г.	Голиб	NJ	Бананы	903	15	13545
3	01 января 2003 г.	Голиб	NJ	Бананы	331	15	4965
4	02 января 2003 г.	Бенавидес	MI	Манго	299	20	5980
5	06 января 2003 г.	Голиб	NJ	Бананы	612	15	9180
6	08 января 2003 г.	Голиб	NJ	Личи	907	12,5	11337,5
7	08 января 2003 г.	Киммел	MI	Рамбутан	107	18	1926
8	10 января 2003 г.	Гомис	CT	Личи	770	12,5	9625
9	14 января 2003 г.	Голиб	NJ	Личи	223	12,5	2787,5
10	14 января 2003 г.	МакДональд	MI	Бананы	132	15	1980
11	15 января 2003 г.	Бенавидес	MI	Личи	669	12,5	8362,5
12	17 января 2003 г.	Бенавидес	MI	Манго	881	15	13215
13	21 января 2003 г.	Гомис	CT	Бананы	624	15	9360
14	22 января 2003 г.	Голиб	NJ	Рамбутан	193	20	3860
15	23 января 2003 г.	МакДональд	MI	Бананы	255	20	5100
16	27 января 2003 г.	Голиб	NJ	Манго	6	20	120

Рис. 19.13. Отчет по объемам продаж

Следующий макрос сравнивает даты с помощью функции VBA `Weekday` для определения дня недели (в виде числа). По умолчанию воскресенье имеет номер 1, а суббота — номер 7. Если номер сегодняшнего дня меньше, чем предыдущего, предполагается, что началась новая неделя и между строками с данными вставляется пустая строка.

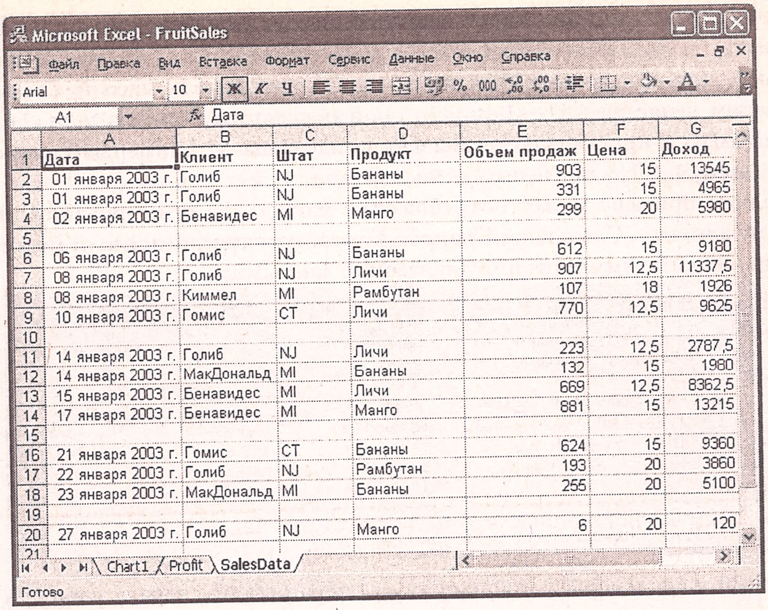
```
Public Sub ShowWeeks()
    Dim Today As Integer
    Dim Yesterday As Integer
    Range("A2").Select
    Yesterday = Weekday(ActiveCell.Value)
    Do Until IsEmpty(ActiveCell.Value)
        ActiveCell.Offset(1, 0).Select
        Today = Weekday(ActiveCell.Value)
        If Today < Yesterday Then
            ActiveCell.EntireRow.Insert

            ActiveCell.Offset(1, 0).Select
        End If
        Yesterday = Today
    Loop
End Sub
```

Результат работы кода показан на рис. 19.14.

Обратите внимание, что многие пользователи определяют пустую ячейку сравнением со строкой нулевой длины.

```
Do Until ActiveCell.Value = ""
```

В большинстве случаев такая проверка срабатывает. Она сработала бы и в предыдущем примере. Но может не сработать для ячеек с формулами, результат которых является строкой нулевой длины. Для обнаружения пустой ячейки лучше использовать функцию VBA `IsEmpty`.



	A	B	C	D	E	F	G
1	Дата	Клиент	Штат	Продукт	Объем продаж	Цена	Доход
2	01 января 2003 г.	Голиб	NJ	Бананы	903	15	13545
3	01 января 2003 г.	Голиб	NJ	Бананы	331	15	4965
4	02 января 2003 г.	Бенавидес	MI	Манго	299	20	5980
5							
6	06 января 2003 г.	Голиб	NJ	Бананы	612	15	9180
7	08 января 2003 г.	Голиб	NJ	Личи	907	12,5	11337,5
8	08 января 2003 г.	Киммел	MI	Рамбутан	107	18	1926
9	10 января 2003 г.	Гомис	CT	Личи	770	12,5	9625
10							
11	14 января 2003 г.	Голиб	NJ	Личи	223	12,5	2787,5
12	14 января 2003 г.	МакДональд	MI	Бананы	132	15	1980
13	15 января 2003 г.	Бенавидес	MI	Личи	669	12,5	8362,5
14	17 января 2003 г.	Бенавидес	MI	Манго	881	15	13215
15							
16	21 января 2003 г.	Гомис	CT	Бананы	624	15	9360
17	22 января 2003 г.	Голиб	NJ	Рамбутан	193	20	3860
18	23 января 2003 г.	МакДональд	MI	Бананы	255	20	5100
19							
20	27 января 2003 г.	Голиб	NJ	Манго	6	20	120

Рис. 19.14. Отчет по объемам продаж с выделением недель пустыми строками

Копирование значений между массивами и диапазонами

Для обработки всех данных диапазона стоит скопировать все значения в массив VBA и обрабатывать массив значений, а не объект `Range`, содержащий значения. После обработки значения можно скопировать обратно в диапазон.

Следующая простая конструкция позволяет скопировать значения из диапазона в массив:

```
SalesData = Range("A2:F10000").Value
```

По сравнению с перебором ячеек данные передаются очень быстро. Обратите внимание, что этот подход отличается от объявления объектной переменной, которая ссылается на диапазон:

```
Set SalesData = Range("A2:F10000")
```

Для присвоения значений диапазона переменной `SalesData` она должна иметь тип `Variant`. Интерпретатор VBA копирует все значения из диапазона в переменную, создавая двумерный массив. Первая размерность соответствует строкам, а вторая — столбцам, поэтому для доступа к значениям в массиве используется номер строки и номер столбца. Для присвоения значения из первой строки и второго столбца массива переменной `Customer` можно воспользоваться следующим кодом:

```
Customer = SalesData(1, 2)
```

Когда значения диапазона присваиваются переменной типа `Variant`, создаваемый массив всегда индексируется начиная с единицы, а не с нуля, независимо от оператора `Option Base` в начале модуля. Кроме этого, массив всегда имеет два измерения, даже если диапазон состоит из одной строки или одного столбца. Структура массива повторя-

ет структуру столбцов и строк листа. Эта особенность помогает при записи массива обратно на лист.

Например, при присвоении значений из диапазона A1:A10 массиву SalesData первым элементом является SalesData(1,1), а последним — SalesData(10,1). Когда массиву SalesData присваиваются значения диапазона A1:E1, первым элементом является SalesData(1,1), а последним — SalesData(1,5).

В последнем примере может потребоваться макрос для суммирования прибыли по конкретному клиенту. Следующий макрос использует традиционный метод непосредственной проверки и суммирования диапазона с данными:

```
Public Sub GoliebToAll()
    Dim Total As Double
    Dim i As Long
    With Range("A2:G20")
        For i = 1 To .Rows.Count
            If .Cells(i, 2) = "Голиб" Then Total =
↳Total + .Cells(i, 7)
        Next i
    End With
    MsgBox "Сумма для Голиба = " & Format(Total, "$#,##0")
End Sub
```

Показанный ниже макрос выполняет ту же операцию, но сначала значения из объекта Range присваиваются переменной типа Variant, после чего обрабатывается полученный массив. Скорость работы макроса заметно увеличивается. Он может работать в пятьдесят раз быстрее, что очень заметно при работе с большими массивами данных.

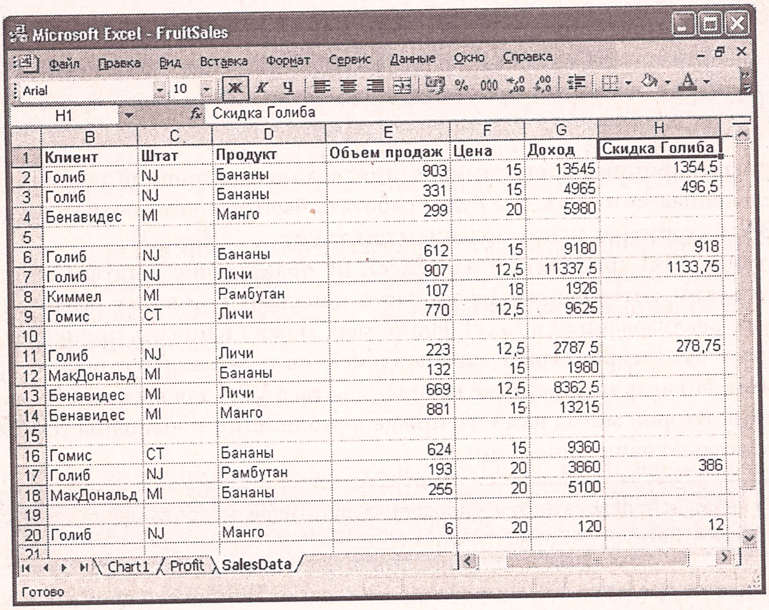
```
Public Sub GoliebTotal2()
    Dim SalesData As Variant
    Dim Total As Double
    Dim i As Long
    SalesData = Range("A2:G20").Value
    For i = 1 To UBound(SalesData, 1)
        If SalesData(i, 2) = "Голиб" Then Total =
↳Total + SalesData(i, 7)
    Next i
    Call MsgBox("Сумма для Голиба  = " & Format(Total, "$#,##0"))
End Sub
```

Кроме этого, массив со значениями можно непосредственно присваивать объекту Range. Предположим, что набор чисел необходимо вывести в столбце H в книге FruitSales.xls из предыдущего примера. Набор чисел является 10% скидкой для клиента "Голиб". В следующем макросе также перед обработкой диапазон значений присваивается переменной типа Variant:

```
Public Sub GoliebDiscount()
    Dim SalesData As Variant
    Dim Discount() As Variant
    Dim i As Long
    SalesData = Range("A2:G20").Value
    ReDim Discount(1 To UBound(SalesData, 1), 1 To 1)
    For i = 1 To UBound(SalesData, 1)
        If SalesData(i, 2) = "Голиб" Then
            Discount(i, 1) = SalesData(i, 7) * 0.1
        End If
    Next i
    Range("H2").Resize(UBound(SalesData, 1), 1).Value = Discount
End Sub
```

Код создает динамический массив, который называется `Discount`. Размерность массива меняется в соответствии с количеством строк в диапазоне `SalesData` (массив имеет соответствующее количество строк и один столбец). После присвоения значений массиву `Discount` этот массив непосредственно присваивается диапазону в столбце H. Обратите внимание, что при этом необходимо правильно указать размер диапазона, в который копируются значения. Указать первую ячейку, как при копировании на листе, недостаточно.

Результат работы этого макроса показан на рис. 19.15.



	B	C	D	E	F	G	H
1	Клиент	Штат	Продукт	Объем продаж	Цена	Доход	Скидка Голиба
2	Голиб	NJ	Бананы	903	15	13545	1354,5
3	Голиб	NJ	Бананы	331	15	4965	496,5
4	Бенавидес	MI	Манго	299	20	5980	
5							
6	Голиб	NJ	Бананы	612	15	9180	918
7	Голиб	NJ	Личи	907	12,5	11337,5	1133,75
8	Киммел	MI	Рамбутан	107	18	1926	
9	Гомис	CT	Личи	770	12,5	9625	
10							
11	Голиб	NJ	Личи	223	12,5	2787,5	278,75
12	МакДональд	MI	Бананы	132	15	1980	
13	Бенавидес	MI	Личи	669	12,5	8362,5	
14	Бенавидес	MI	Манго	881	15	13215	
15							
16	Гомис	CT	Бананы	624	15	9360	
17	Голиб	NJ	Рамбутан	193	20	3860	386
18	МакДональд	MI	Бананы	255	20	5100	
19							
20	Голиб	NJ	Манго	6	20	120	12

Рис. 19.15. Копирование массива в диапазон ячеек

Массив `Discount` может быть и одномерным. Но во время присвоения диапазону одномерного массива он будет содержать строку данных, а не столбец. Для обхода этого ограничения можно воспользоваться функцией листа `Transpose`. Предположим, что размерности массива `Discount` изменены следующим образом:

```
ReDim Discount(1 To Ubound(SalesData,1))
```

Эту версию массива можно присвоить столбцу с помощью такой конструкции:

```
Range("H2").Resize(UBound(SalesData, 1), 1).Value = _
   WorkSheetFunction.Transpose(vaDiscount)
```

Удаление строк

Достаточно часто у разработчиков возникает вопрос: "Как лучше всего удалить строки листа, которые не потребуются в дальнейшем?" Обычно необходимо найти и удалить строки, содержащие определенный текст в определенном столбце. Наиболее подходящее решение зависит от размера электронной таблицы и ориентировочного количества удаляемых элементов.

Предположим, необходимо удалить все строки, содержащие текст "Mangoes" в столбце D. Одним из способов является перебор строк в цикле и проверка содержимого каждой ячейки в столбце D. В таком случае лучше начинать с последней строки и перемещаться вверх строка за строкой. При этом Excel не нужно передвигать вверх строки, которые потом придется удалить. Если начать сверху не получится, то лучше использовать простой цикл `For... Next`, так как при удалении строк значение счетчика цикла не будет соответствовать номеру строки.

```
Public Sub DeleteRows1()
    Dim i As Long
    Application.ScreenUpdating = False
    For i = Cells(Rows.Count, "D").End(xlUp).Row To 1 Step -1
        If Cells(i, "D").Value = "Mangoes" Then Cells(i,
➥"D").EntireRow.Delete
    Next i
End Sub
```

Есть хороший принцип программирования: если в электронной таблице Excel реализуется определенная методика, она будет работать быстрее, чем соответствующая реализация на VBA, например, с использованием цикла `For... Next`.

Разработчики VBA-приложений для Excel, не обладающие богатым опытом работы с пользовательским интерфейсом Excel, часто попадают в ловушку реализации на VBA операции, уже реализованной в Excel. Например, можно написать процедуру VBA, которая перебирает отсортированный список элементов и вставляет строки с промежуточными суммами. Но ведь можно также воспользоваться VBA для вызова метода `Subtotal` объекта `Range`. Второй метод намного проще в реализации и выполняется быстрее, чем тело самостоятельно написанного цикла.

> Язык VBA лучше использовать для управления встроенными возможностями Excel, чем для повторной реализации функциональности Excel.

Но подходящее средство Excel не всегда очевидно. Наиболее вероятным кандидатом на поиск удаляемых ячеек без просмотра каждой строки является команда Правка⇨Найти (Edit⇨Find). В следующем коде метод `Find` используется для сокращения итераций цикла VBA:

```
Public Sub DeleteRows2()
    Dim FoundCell As Range
    Application.ScreenUpdating = False
    Set FoundCell = Range("D:D").Find(what:="Mangoes")
    Do Until FoundCell Is Nothing
        FoundCell.EntireRow.Delete
        Set FoundCell = Range("D:D").FindNext
    Loop
End Sub
```

При небольшом количестве удаляемых строк этот код работает быстрее, чем первая процедура. С увеличением относительной доли удаляемых строк реализация становится менее эффективной. Возможно, на определенном этапе придется искать другое средство Excel.

Самым быстрым известным способом удаления строк является использование возможности Автофильтр (AutoFilter).

```
Public Sub DeleteRows3()
    Dim LastRow As Long
    Dim aRange As Range
```

```
    Application.ScreenUpdating = False
    Rows(1).Insert
    Range("D1").Value = "Temp"
    With ActiveSheet
        .UsedRange
        LastRow = .Cells.SpecialCells(xlCellTypeLastCell).Row
        Set aRange = Range("D1", Cells(LastRow, "D"))
        aRange.AutoFilter Field:=1, Criteria1:="Манго"
        aRange.SpecialCells(xlCellTypeVisible).EntireRow.Delete
        .UsedRange
    End With
End Sub
```

Этот способ сложнее в реализации, но намного быстрее в работе и не зависит от количества удаляемых строк. Для использования свойства AutoFilter в верхней строке диапазона с данными должны указываться имена полей. Сначала над диапазоном с данными вставляется строка с именем для столбца D. Свойство AutoFilter используется только по отношению к столбцу D. При этом скрываются все строки, не содержащие строку "Манго".

Метод SpecialCells применяется для выделения только видимых строк в столбце D. Это выделение распространяется на всю видимую строку, и строки удаляются включая строку с названием поля. Свойства AutoFilter автоматически отключается при удалении строки с названием поля.

Резюме

В этой главе рассматривались самые важные свойства и методы, позволяющие работать с диапазонами ячеек на листе электронной таблицы. Основное внимание уделялось методикам, которые невозможно выявить при использовании механизма записи макросов. Рассматривались, в частности, следующие свойства и методы:

- метод Activate;
- свойство Cells;
- свойства Columns и Rows;
- свойство CurrentRegion;
- свойство End;
- свойство Offset;
- свойство Range;
- свойство Resize;
- метод Select;
- метод SpecialCells;
- методы Union и Intersect.

В главе было показано, как присваивать значения из диапазона на листе массиву VBA и как присваивать массив диапазону листа. Перемещение значений в массив позволяет ускорить обработку.

Также в этой главе отмечалось, что необходимость в активизации или выделении ячеек возникает очень редко, хотя при использовании механизма записи макросов эти

выполняемые вручную операции записываются всегда. Активизация ячеек и листов занимает очень много времени. Для обеспечения максимальной производительности кода этого процесса стоит избегать.

Из последних примеров следовало, что обычно лучше использовать существующие возможности Excel, предоставленные объектной моделью, чем создавать собственный эквивалент этой функциональности на языке VBA. Не забывайте, что некоторые приемы использования Excel более эффективны. Для получения максимального быстродействия может потребоваться несколько экспериментов.

20

Использование имен

Одной из наиболее полезных возможностей Excel является создание имен. Их можно создавать с помощью команды меню Вставка⇨Имя⇨Присвоить (Insert⇨Name⇨Define). Для создания имени диапазона необходимо выделить диапазон, ввести имя в поле Имя (Name) в левой части панели Формула (Formula) и нажать клавишу <Enter>. Но в Excel имена могут ссылаться не только на диапазоны.

Имя может содержать число, текст или формулу. Такое имя не имеет видимого положения на листе и может просматриваться только в диалоговом окне Вставка⇨Имя⇨Присвоить (Insert⇨Name⇨Define). Таким образом, имена можно использовать для хранения информации в книге без добавления данных в ячейки листа. Имена могут быть объявлены скрытыми. В таком случае имя не отображается в диалоговом окне Вставка⇨Имя⇨Присвоить (Insert⇨Name⇨Define). Это один из способов сокрытия полезной информации от пользователей.

Обычно имена применяются для слежения за диапазонами листов. Имена оказываются особенно полезными при использовании таблиц переменного размера для хранения данных. Если известно, что определенное имя ссылается на диапазон с обрабатываемыми данными, можно применять более простой код VBA. Кроме этого, существует несколько простых приемов изменения определения имени, позволяющих модифицировать таблицы с данными с помощью кода VBA.

В составе объектной модели Excel предоставляется коллекция `Names` и объект `Name`, используемые в коде VBA. Имена могут быть определены глобально на уровне книги, а могут быть локальными на уровне листа. При создании локальных имен одно и то же имя может использоваться в нескольких листах книги. Для создания объекта `Name` на уровне листа перед значением свойства `Name` указывается имя листа и восклицательный знак. Например, для определения локального в пределах листа `Sheet1` имени `Costs` в диалоговом окне Вставка⇨Имя⇨Присвоить (Insert⇨Name⇨Define) можно ввести **Sheet1!Costs** (рис. 20.1).

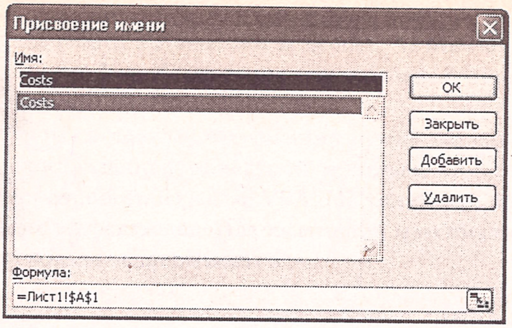


Рис. 20.1. Диалоговое окно, доступное по команде Вставка⇨Имя⇨Присвоить

При выводе диалогового окна Вставка⇨Имя⇨Присвоить (Insert⇨Name⇨Define) отображаются глобальные имена из книги и локальные имена из активного листа. Локальные имена квалифицируются именем листа справа. Если локальное имя совпадает с глобальным, отображается только локальное имя.

Одним из источников путаницы является наличие имен у имен. Объект `Name` и свойство `Name` этого объекта различаются. Следующий код возвращает ссылку на объект `Name` из коллекции `Names`:

```
Names("Costs")
```

Для изменения свойства `Name` объекта `Name` можно воспользоваться таким кодом:

```
Names("Costs").Name = "NewData"
```

В результате изменения свойства `Name` к объекту `Name` можно обращаться следующим образом:

```
Names("NewData")
```

Глобальные и локальные имена хранятся в коллекции `Names`, связанной с объектом `Workbook`. При использовании записи вида `Application.Names` или `Names` возвращается ссылка на коллекцию `Names` активной книги. Для получения ссылки на коллекцию `Names` конкретной книги можно воспользоваться следующим кодом:

```
Workbooks("Book1.xls").Names
```

Локальные (но не глобальные) имена также принадлежат коллекции `Names`, связанной с соответствующим объектом `Worksheet`. Для обращения к локальной коллекции `Names` интересующего листа можно воспользоваться ссылкой вида `Worksheets("Sheet1").Names` или `ActiveSheet.Names`.

Существует еще один способ обращения к именам, описывающим диапазоны. Для этого можно использовать свойство `Name` объекта `Range`. Дополнительная информация об этом способе приводится ниже.

Именование диапазонов

Для создания ссылающегося на диапазон глобального имени можно воспользоваться методом `Add` коллекции `Names` из объекта `Workbook`.

```
Names.Add Name:="Data", RefersTo:="=Sheet1!$D$10:$D$12"
```

Важно, чтобы перед определением указывался символ равенства и применялись абсолютные ссылки на ячейки (абсолютные ссылки начинаются с символа $). В противном случае имя будет соответствовать адресу относительно активной ячейки на момент определения имени. Если имя должно соответствовать активному листу, ссылку на лист можно опустить.

```
Names.Add Name:="Data", RefersTo:="=$D$10:$D$12"
```

Если имя уже существует, существующее определение будет заменено новым.

Следующий код позволяет создать локальное имя:

```
Names.Add Name:="Sheet1!Sales", RefersTo:="=Sheet1!$E$10:$E$12"
```

С другой стороны, имя можно добавить в коллекцию Names, связанную с интересующим листом. В этой коллекции хранятся только локальные имена листа:

```
Worksheets("Sheet1").Names.Add Name:="Costs",
↳RefersTo:="=Sheet1!$F$10:$F$12"
```

Использование свойства Name объекта Range

Существует более простой способ создания имени для объекта Range. Для этого можно обратиться к свойству Name объекта Range.

```
Range("A1:D10").Name = "SalesData"
```

Если имя должно быть локальным, в значение свойства можно добавить имя листа:

```
Range("F1:F10").Name = "Sheet1!Staff"
```

При создании кода с объектами Range проще работать таким образом, чем генерировать строку адреса диапазона после символа равенства (эта строка передается в качестве значения параметра RefersTo метода Add коллекции Names). Например, если после создания объектной переменной aRange ей необходимо присвоить имя Data, значение свойства Address объекта aRange придется присоединить к строке вызова, как показано ниже:

```
=:Names.Add Name:="Data", RefersTo:="=" & aRange.Address
```

С другой стороны, можно воспользоваться следующим кодом:

```
aRange.Name = "Data"
```

Но полностью забыть о методе Add не получится, так как это единственный способ создания имен для чисел, формул и строк.

Специальные имена

Некоторые имена используются внутри Excel для отслеживания определенных возможностей. При выборе диапазона печати на листе Excel присваивает этому диапазону локальное имя Print_Area. При создании заголовков печати Excel создает локальное имя Print_Titles, а при извлечении данных из списка в новый диапазон с помощью команды Данные⇨Фильтр⇨Расширенный фильтр (Data⇨Filter⇨Advanced Filter) Excel создает локальные имена Criteria и Extract.

В более старых версиях Excel имя Database *использовалось для диапазонов с данными. Хотя в современных версиях имя* Database *применять не обязательно, это имя все еще распознается некоторыми компонентами Excel, например* Расширенным фильтром *(Advanced Filter).*

Макрос, применяющий для редактирования списка данных команду Данные⇨Форма (Data⇨Form), не будет работать со списками, которые начинаются за пределами диапазона A1:B2. Для обхода этого ограничения списку данных можно присвоить имя Database.

О применении этих имен в Excel стоит знать и избегать их использования в собственных приложениях (кроме случаев специального получения побочных эффектов, связанных с этими именами). Например, для удаления области печати можно удалить имя Print_Area. При наличии определенной области печати следующие две строки кода являются эквивалентными:

```
ActiveSheet.PageSetup.PrintArea = ""
ActiveSheet.Names("Print_Area").Delete
```

Как итог, можно отметить следующие имена, при работе с которыми необходимо соблюдать осторожность:

- Criteria;
- Database;
- Extract;
- Print_Area;
- Print_Titles.

Хранение значений в именах

Использование имен для хранения элементов данных уже рассматривалось в главе 3 в разделе "Метод Evaluate". Теперь пришло время более подробно проанализировать эту возможность.

При использовании имени для хранения числовых или строковых данных перед значением параметра RefersTo не нужно указывать символ =. Если это сделать, значение будет рассматриваться как формула. Следующий код сохраняет число и строку в именах StoreNumber и StoreString соответственно:

```
Dim X As Variant
X = 3.14159
Names.Add Name:="StoreNumber", RefersTo:=X
X = "Sales"
Names.Add Name:="StoreString", RefersTo:=X
```

Эта возможность очень удобна при хранении данных для кода VBA между сеансами работы в Excel. Данные не исчезают при закрытии Excel, и поддерживается хранение строк длиной до 255 символов.

Для получения значения имени можно воспользоваться методом Evaluate.

```
X = [StoreNumber]
```

Кроме этого, в именах можно хранить формулы. Формула должна начинаться с символа =. Следующая строка кода сохраняет в имени вызов функции COUNTA.

```
Names.Add Name:="ItemsInA", RefersTo:="=COUNTA($A:$A)"
```

Это имя может использоваться в формулах в ячейках листа для получения количества элементов в столбце A (рис. 20.2).

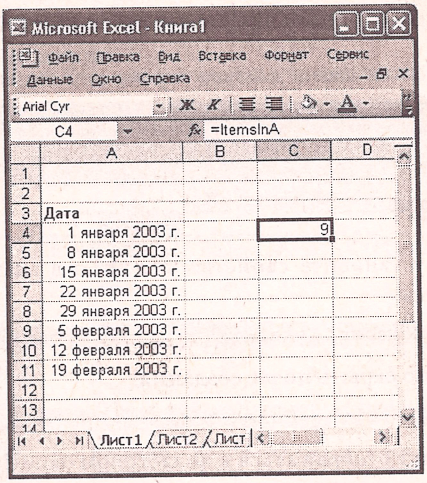


Рис. 20.2. Использование имен для хранения формулы листа

И в этом случае для получения значения имени в коде VBA можно воспользоваться методом `Evaluate`:

```
MsgBox [ItemsInA]
```

Хранение массивов

В имени можно хранить переменную массива с набором значений. Следующий код создает массив чисел `MyArray` и сохраняет значения массива под именем `MyName`.

```
Public Sub ArrayToName()
    Dim MyArray(1 To 200, 1 To 3)
    Dim I As Integer
    Dim J As Integer

    For I = 1 To 200
        For J = 1 To 3
            MyArray(I, J) = I + J
        Next J
    Next I
    Names.Add Name:="MyName", RefersTo:=MyArray
End Sub
```

> В Excel 2003 размер массива ограничен только размером памяти и дискового пространства, а также воображением разработчика.

Метод `Evaluate` может использоваться для присвоения значений содержащего массив имени переменной типа `Variant`. Следующий код присваивает содержимое имени `MyName` (создается в подпрограмме `ArrayToName`) переменной `MyArray`. После этого выводится последний элемент массива:

```
Public Sub NameToArray()
    Dim MyArray As Variant
    MyArray = [MyName]
    MsgBox MyArray(200, 3)
End Sub
```

В результате присвоения имени массива переменной типа `Variant` *всегда создается массив с нумерацией начиная с 1, даже если в начале модуля присутствует оператор* `Option Base 0`.

Сокрытие имен

Для сокрытия имени установите свойство `Visible` в значение `False`. Это можно сделать при создании имени:

```
Names.Add Name:="StoreNumber", RefersTo:=x, Visible:=False
```

Кроме этого, имя можно скрыть после создания:

```
Names("StoreNumber").Visible = False
```

После этого имя не будет отображаться в диалоговом окне Вставка⇨Имя⇨Присвоить (Insert⇨Name⇨Define). Этого недостаточно для безопасного сокрытия информации, так как VBA позволяет легко обнаружить такое имя, но это эффективный способ защиты пользователей от странных имен.

Кроме этого, стоит помнить, что если через пользовательский интерфейс Excel будет создано имя, совпадающее со скрытым именем, скрытое имя будет уничтожено. В этом случае для сохранения имен стоит защитить лист.

Несмотря на ряд ограничений, скрытые имена являются неплохим хранилищем информации в пределах книги.

Работа с именованными диапазонами

На показанном ниже листе (рис. 20.3) приводится список данных B4:D10, которому присвоено имя `Database`. Кроме этого, в диапазоне B2:D2 находится область ввода данных, которой присвоено имя `Input`.

Следующий код позволяет скопировать данные из диапазона `Input` в диапазон со списком данных и переопределить имя `Database` для включения добавленных данных:

```
Public Sub AddNewData()
    Dim Rows As Long

    With Range("Database")
        Rows = .Rows.Count + 1
        Range("Input").Copy Destination:=.Cells(Rows, 2)
        .Resize(Rows).Name = "Database"
    End With
End Sub
```

В результате работы этого кода в диапазон B11:D11 будут скопированы данные “Shelley 26 Ж”. После этого имя `Database` будет ссылаться на диапазон B4:D11.

В переменной `Rows` хранится счетчик количества строк в диапазоне `Database` плюс еще одна строка для записи новых данных. После этого копируется диапазон `Input`. Данные копируются в ячейку B11, которая определена через свойство `Cells` объекта

Database. Интересующая ячейка находится в первом столбце диапазона Database. Расстояние от вершины диапазона до интересующей ячейки хранится в переменной Rows. Свойство Resize объекта Database возвращает ссылку на объект Range, в котором на одну строку больше, чем в диапазоне Database. Свойству Name нового объекта присваивается значение Database.

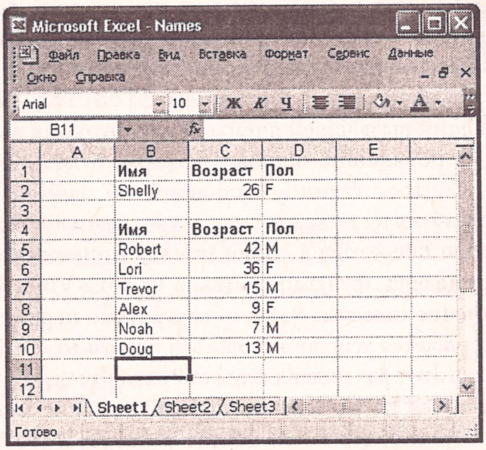


Рис. 20.3. Диапазон со списком данных и область ввода

Приятной особенностью этого кода является независимость от размера и положения диапазона Database на активном листе и от положения диапазона Input в пределах активной книги. Диапазон Database может состоять из семи или семи тысяч строк. В диапазоны Input и Database можно добавить дополнительные столбцы и код будет работать точно так же. Он будет работать, даже если диапазоны Input и Database находятся на разных листах книги.

Поиск имени

Если необходимо проверить существование имени в пределах книги, можно воспользоваться следующей функцией. Эта функция может применяться как в качестве функции листа, так и в качестве функции VBA. В результате она оказалась немного сложнее, чем могла бы быть.

```
Public Function IsNameInWorkbook(ByVal Name As String) As Boolean
    Dim X As String
    Dim aRange As Range

    Application.Volatile
    On Error Resume Next
    Set aRange = Application.Caller
    Err.Clear

    If aRange Is Nothing Then
        X = ActiveWorkbook.Names(Name).Name
    Else
        X = aRange.Parent.Parent.Names(Name).Name
```

```
    End If

    If Err.Number = 0 Then IsNameInWorkbook = True
End Function
```

Функция IsNameInWorkbook принимает аргумент Name, в котором в виде строки передается интересующее имя. Функция определена как непостоянная, поэтому значение функции листа пересчитывается каждый раз при добавлении или удалении интересующего имени. Сначала функция определяет источник вызова. Для этого значение свойства Application.Caller присваивается переменной aRange.

Если функция вызвана из ячейки, свойство Application.Caller возвращает объект Range, соответствующий вызвавшей ячейке. Если функция вызывалась не из ячейки, оператор Set приведет к появлению ошибки, которая будет подавлена оператором On Error Resume Next. Соответственно, сообщение об ошибке удаляется, так как следующие сообщения об ошибках не должны скрываться этим сообщением.

После этого с помощью оператора If проверяется определение переменной aRange. Если переменная не определена, функция вызвана из кода VBA. В таком случае функция пытается присвоить значение свойства Name объекта Name переменной X. Если такое имя существует, попытка завершается успешно и ошибка не возникает. В противном случае возникает ошибка, которая и в этот раз подавляется оператором On Error Resume Next.

Если функция вызывается из ячейки листа, предложение Else оператора If идентифицирует книгу, в которой содержится объект aRange, и пытается присвоить значение свойства Name объекта Name переменной X. Родительским объектом для aRange является лист, в котором хранится объект aRange. Родительским объектом для листа является книга, в которой хранится объект aRange. И в этот раз, если имя не существует, возникает ошибка.

Наконец, функция IsNameInWorkbook сравнивает с нулем значение свойства Number объекта Err. Если значение равно нулю, функция возвращает значение True (имя существует). Если значение не равно нулю, функция возвращает значение False (имя не существует).

Ниже показано использование функции IsNameInWorkbook в ячейке электронной таблицы.

```
=IF(IsNameInWorkbook("Lori"),"Имя Lori ","Имя Lori не ")&"существует"
```

Следующая подпрограмма запрашивает имя у пользователя и проверяет существование этого имени.

```
Sub TestName()
    If IsNameInWorkbook(InputBox("What Name")) Then
        MsgBox "Имя существует"
    Else
        MsgBox "Имя не существует"
    End If
End Sub
```

Обратите внимание, что при поиске локального имени необходимо указывать имя листа. Для этого используется форма записи вида Sheet1!Name.

При вызове функции IsNameInWorkbook в качестве функции листа, если имя "Lori" существует, будет выдано следующее сообщение (рис. 20.4).

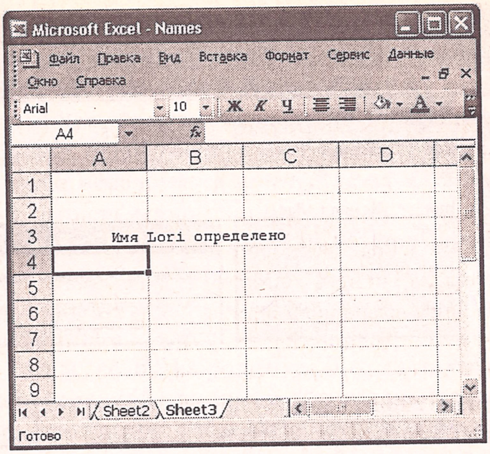


Рис. 20.4. Результат проверки существования имени

Поиск имени диапазона

Если диапазону присвоено имя и свойство `RefersTo` объекта `Name` точно соответствует координатам диапазона, то свойство `Name` объекта `Range` возвращает имя диапазона.

Может возникнуть соблазн воспользоваться данным кодом для вывода имени диапазона:

```
MsgBox aRange.Name
```

Этот код не работает, так как свойство `Name` объекта `Range` возвращает объект `Name`. Предыдущий код выводит значение принятого по умолчанию свойства объекта `Name` (это свойство `RefersTo`). Для вывода значения свойства `Name` объекта `Name` необходимо воспользоваться следующим кодом:

```
MsgBox aRange.Name.Name
```

Этот код работает только в том случае, если диапазону `aRange` присвоено имя. Если имя не присваивалось, выполнение кода приводит к ошибке времени выполнения. Следующий код позволяет вывести имя выделенной ячейки на активном листе:

```
Public Sub TestNameOfRange()
    Dim aName As Name
    On Error Resume Next
    Set aName = Selection.Name
    If aName Is Nothing Then
        MsgBox "Выделению не присвоено имя"
    Else
        MsgBox "Выделению присвоено имя " & aName.Name
    End If
End Sub
```

Если диапазону `aRange` присвоено несколько имен, выводится имя, первое в алфавитном порядке.

Результат работы этого макроса показан на рис. 20.5.

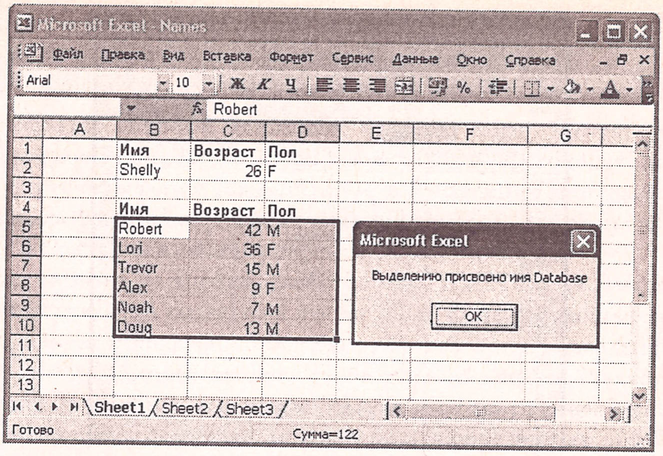


Рис. 20.5. Определение имени выделенного диапазона

Определение имен, пересекающих диапазон

При проверке присвоенных диапазонам листа имен часто возникает необходимость определить все имена, которые связаны с выделенными ячейками. Иногда интересуют имена, которые полностью покрывают выделенные ячейки, а иногда достаточно знать частичное пересечение имен с выделением. Следующий код выводит список всех имен, которые полностью покрывают выделенные ячейки активного листа:

```
Public Sub SelectionEntirelyInNames()
     Dim Message As String
    Dim aName As Name
    Dim NameRange As Range
    Dim aRange As Range

    On Error Resume Next

    For Each aName In Names
        Set NameRange = Nothing
        Set NameRange = aName.RefersToRange
        If Not NameRange Is Nothing Then
            If NameRange.Parent.Name = ActiveSheet.Name Then
                Set aRange = Intersect(Selection, NameRange)
                If Not aRange Is Nothing Then
                    If Selection.Address = aRange.Address Then
                        Message = Message & aName.Name & vbCr
                    End If
                End If
            End If
        End If
    Next aName
    If Message = "" Then
        MsgBox "Ни одно имя не покрывает выделение полностью"
    Else
        MsgBox Message
    End If
End Sub
```

В начале подпрограммы SelectionEntirelyInNames подавляется вывод сообщений об ошибках (On Error Resume Next). После этого начинается цикл For Each... Next, который обрабатывает все имена в пределах книги. Между итерациями цикла значение переменной NameRange устанавливается равным Nothing, что позволяет удалить ссылку на диапазон, оставшуюся от последней итерации. После этого свойство Name текущего объекта Name используется в качестве объекта Range и ссылка на этот объект Range присваивается переменной NameRange. Если имя не ссылается на диапазон, то операция завершается неудачно, поэтому остаток цикла выполняется только в случае присвоения переменной NameRange действительного объекта Range.

Следующий оператор If проверяет попадание ссылки NameRange в активный лист. Содержащий переменную NameRange лист является родительским. Внутренний код выполняется только в том случае, если имя родительского листа совпадает с именем активного листа. После этого переменной Rng присваивается пересечение выделенных ячеек и диапазона NameRange. Если пересечение существует и переменная aRange не равна Nothing, выполняется самый внутренний оператор If.

В последнем операторе If проверяется идентичность диапазона пересечения в переменной aRange и выделенного диапазона. Если диапазоны идентичны, выделенные ячейки полностью входят в диапазон NameRange и свойство Name текущего объекта Name добавляется к именам, которые уже указаны в строке Message. Кроме этого, добавляется символ возврата каретки (с помощью встроенной константы vbCr), чтобы каждое имя выводилось в новой строке сообщения.

После завершения работы цикла For Each... Next оператор If проверяет наличие строки в переменной Message. Если переменная Message хранит строку нулевой длины, с помощью функции MsgBox выводится сообщение о том, что имена не найдены. Иначе выводится список имен, перечисленных в строке Message.

При запуске подпрограммы SelectionEntirelyInNames в пределах электронной таблицы с тремя именованными диапазонами (Data, Fred и Mary) будет получен результат, показанный на рис. 20.6.

Если необходимо определить имена, только частично пересекающие выделение, можно удалить второй внутренний оператор If (в результате будет получен следующий код):

```
Public Sub NamesOverlappingSelection()
    Dim Message As String
    Dim aName As Name
    Dim NameRange As Range
    Dim aRange As Range

    On Error Resume Next
    For Each aName In Names
        Set NameRange = Nothing
        Set NameRange = Range(aName.Name)
        If Not NameRange Is Nothing Then

            If NameRange.Parent.Name = ActiveSheet.Name Then
                Set aRange = Intersect(Selection, NameRange)
                    If Not aRange Is Nothing Then
                        Message = Message & aName.Name & vbCr
                    End If
            End If
        End If
    Next aName
    If Message = "" Then
        MsgBox "Ни одно из имен не пересекает выделение"
```

```
    Else
        MsgBox Message
    End If
End Sub
```

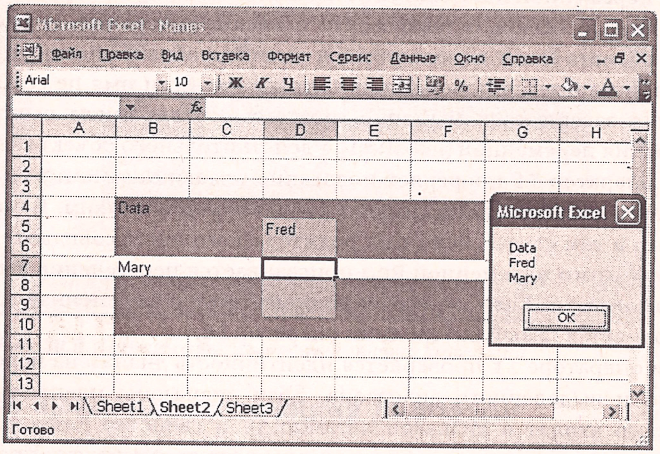


Рис. 20.6. Определение имен, покрывающих выделение

Обратите внимание, что в подпрограммах SelectionEntirelyInNames и NamesOverlappingSelection используются различные способы присвоения объектной переменной rgNameRange диапазона, на который ссылается имя. Следующие операторы являются эквивалентными:

```
Set NameRange = Name.RefersToRange
Set NameRange = Range(Name.Name)
```

Резюме

В этой главе подробно рассматривались механизмы работы с именами в Excel VBA. Было показано, как использовать имена для слежения за диапазонами листов, хранения числовых и строковых данных. Кроме этого, рассматривалась возможность сокрытия имен, проверки существования имени в пределах книги и диапазона, а также определение частичного или полного пересечения именованных диапазонов. Эти навыки позволяют управлять сложными связями между перекрывающимися и пересекающимися данными.

21

Работа со списками

Список элементов часто используется в качестве абстракции реального мира. Например, люди создают списки действий, покупок, акций, мест, передач, книг и т.д. В этой главе рассматриваются списки, отношения между списками и диапазонами, а также публикация и совместное использование списков в сети Internet.

Создание списка

Список похож на диапазон с дополнительными возможностями. При создании списка на основе диапазона ячеек Excel добавляет автоматический фильтр в начале списка. Этот фильтр поддерживает сортировку и фильтрацию списка, обеспечивает точку вставки и предоставляет маркер изменения размера в нижней правой части списка для ручного изменения размеров списка.

Excel предоставляет очень простую процедуру создания списка. Щелкните в любом месте непрерывного диапазона ячеек и нажмите комбинацию клавиш <Ctrl+L> или выберите команду Данные⇨Список⇨Создать список (Data⇨List⇨Create List). В результате откроется диалоговое окно Создание списка (Create List) (рис. 21.1). В диалоговом окне выводится достаточно точное предположение о содержимом списка. Установите флажок Список с заголовками (My list has headers), если в первой строке списка содержится информация о заголовках. После щелчка на кнопке OK будет создан список с раскрывающимся списком фильтра в верхней части каждого столбца и областью ввода в нижней части каждого столбца.

Каждый из описанных элементов (кроме итоговых значений) показан на рис. 21.2. Раскрывающийся список фильтра в начале списка позволяет выполнять сортировку в порядке возрастания и убывания по определенному элементу списка. Список выделен темно-синей границей. Для сокрытия границы неактивного листа можно выбрать команду Данные⇨Список⇨Скрывать границы неактивных списков (Data⇨List⇨Hide Border of

Inactive Lists). Маркер изменения размера работает, как и любой другой элемент управления размера. Маркер обеспечивает выравнивание по границам столбца и строки. Символ (*) указывает положение точки ввода. Итог для списка создается командой Данные⇨Список⇨Строка итогов (Data⇨List⇨Total Row).

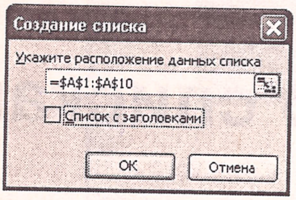


Рис. 21.1. Создание списка

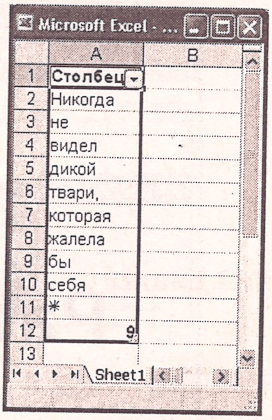


Рис. 21.2. Результат создания списка

Сокращенные команды для списков

Как и большинство возможностей, списки добавляют контекстные меню с характерными для списков командами. Панель инструментов Список (Lists) и контекстное меню списков содержат команды для вставки и удаления столбцов и строк, удаления содержимого списка, преобразования списка в диапазон, суммирования, сортировки и изменения размера. При щелчке правой кнопкой мыши на списке выводится контекстное меню. В данный момент интерес вызывают меню Вставить (Insert) и Удалить (Delete). В каждом меню доступны пункты Строка (Row) и Столбец (Column), которые позволяют вставить или удалить столбец или строку.

Сортировка и фильтрация списка

Обычно для донесения дополнительного смысла можно изменять порядок и форму записи данных. Например, числа 11, 5, 3, 7 и 2 могут выглядеть последовательностью случайных чисел, но после сортировки в порядке возрастания (2, 3, 5, 7, 11) становится ясно, что это последовательность простых чисел.

Размещение элементов в контексте часто позволяет сообщить дополнительный смысл. Организация (сортировка и фильтрация) является еще одним методом передачи информации. Возможность Автофильтр (AutoFilter) позволяет сортировать и фильтровать данные списков. На рис. 21.3 показаны доступные по умолчанию параметры сортировки. Например, при создании индекса может потребоваться сортировка в алфавитном порядке.

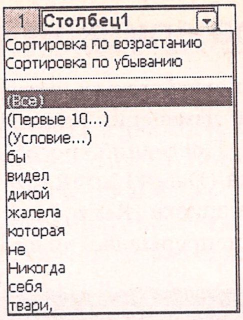


Рис. 21.3. Доступные параметры сортировки списка

Создание диалогового окна UserForm на основе списка

Автоматическая генерация диалоговых окон лежит в основе программирования на Visual Basic. Всего десяток лет назад создание простого диалогового окна для ввода данных требовало значительных трудозатрат. Диалоговое окно рисовалось с помощью символов ANSI. Значение автоматической генерации пользовательского интерфейса сложно переоценить. Если необходимо просматривать список в виде диалогового окна UserForm, выделите список и выберите пункт Данные⇨Форма (Data⇨Form). Как показано на рис. 21.4, эта команда позволяет получить полностью функциональное диалоговое окно практически без дополнительных трудозатрат.

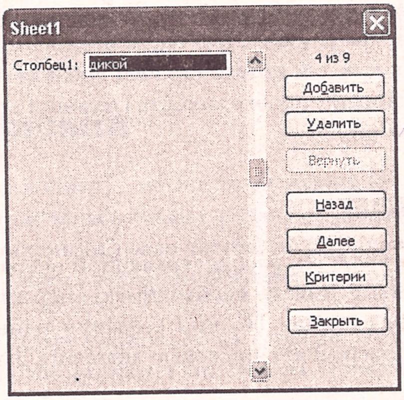


Рис. 21.4. Диалоговое окно, сгенерированное автоматически

Выше показано, как можно определить диалоговое окно, позволяющее добавлять, удалять и фильтровать данные. (Можно отметить, что в начале 90-х годов прошлого века приходилось заниматься ламинированием ручных аппликаций графического интерфейса, похожего на графический интерфейс, который показан на рис. 21.4. После этого несколько дней уходило на его реализацию.)

Изменение размера списков

После создания списка может потребоваться изменение его размеров. Для этого можно воспользоваться пиктограммой изменения размера в нижнем левом углу списка или указать граничные ячейки списка. Последний способ доступен через команду Данные⇨Список⇨Изменить размер списка (Data⇨List⇨Resize List). После этого откроется диалоговое окно Изменение размера списка (Resize List) (рис. 21.5). В этом диалоговом окне список можно преобразовать в непрерывный диапазон ячеек.

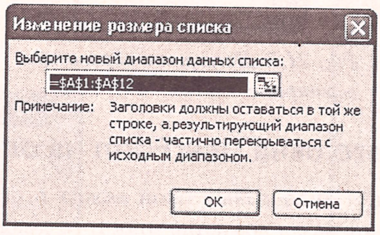


Рис. 21.5. Диалоговое окно Изменение размера списка

Перетаскивание маркера изменения размера в нижнем углу списка

Как было показано ранее, пользователь может изменить размер списка с помощью перетаскивания пиктограммы изменения размера. В большинстве случаев это самый простой способ изменения размера списка. При таком изменении размера в список добавляются строки и столбцы, а в каждый новый столбец добавляется заголовок автофильтра. При этом крайне редко возникает необходимость создания списка на основе нескольких листов или из несмежных ячеек.

Суммы столбцов

Подсчет суммы столбца является распространенной операцией. Стоит отметить, что Excel пытается самостоятельно угадать необходимую операцию и просто подсчитывает количество строк в каждом столбце. Но при создании одинакового количества строк с простыми числами Excel распознает числовые значения и подсчитает сумму простых чисел в списке.

Для подсчета суммы достаточно щелкнуть на кнопке Переключить строку итогов (Toggle Total Row) на панели инструментов Список (List) (рис. 21.6).

Рис. 21.6. Панель инструментов Список (List)

Преобразование списка в диапазон

Команда меню Данные⇨Список⇨Преобразовать в диапазон (Data⇨List⇨Convert to Range) выполняет преобразование списка в диапазон. Обычно создается большой объем кода для работы с диапазонами. Вместо отказа от этого кода лучше программно преобразовать список в диапазон. В таком случае усилия по созданию кода для работы с диапазонами не пропадут даром. В следующем листинге показано использование списка в качестве диапазона (для этого выводятся ячейки, входящие в список/диапазон).

```
Public Sub ListToRange()

    Dim List As ListObject
    Set List = Sheet1.ListObjects(1)

    Dim R As Range
    Set R = List.DataBodyRange

    MsgBox R.Cells.Address

End Sub
```

Свойство `ListObjects` объекта `Worksheet` возвращает коллекцию списков в пределах конкретного листа. Эта возможность может использоваться вне конкретного листа. Имя интересующего листа передается в качестве параметра. Метод `List.DataBodyRange` возвращает ячейки, составляющие диапазон. Если интересует часть списка, представленная в виде диапазона, передайте номера строк и столбцов в качестве параметров метода `DataBodyRange`. С помощью этого метода можно получить ссылку на диапазон, что обеспечивает возможность использования существующего кода, предназначенного для работы с диапазонами.

Публикация списков

“Мыльный пузырь .com лопнул”*, но никто не сказал об этом компаниям eBay.com и Amazon.com. Эти компании зарабатывают реальные деньги в сети Internet. Хотя подобное удается не всем компаниям, мало компаний отрицают полезность присутствия в Web. Внутренние сайты, сайты электронной коммерции и распространенность данных добавляют сил.

Интересующие списки могут храниться на сервере, работающем под управлением Microsoft SharePoint. После копирования списков на сервер Excel позволяет обновлять, синхронизировать и удалять списки.

Предположим, отец игрока детской хоккейной команды хочет предоставить контактную информацию другим родителям игроков команды. Создание списка с информацией и его публикация на сайте SharePoint не займет много времени. Если кто-то сменит номер телефона, разведется или уйдет работать в другую компанию, этот человек сможет просмотреть список и обновить контактную информацию. Старая привычка записывать имена друзей на обратной стороне телефонной книги может превратиться в привычку записывать имена на цифровом носителе.

---

* *Первый инвестиционный бум в Internet закончился полным крахом. Из тысяч компаний остались единицы.* — Примеч. ред.

Для публикации списка данных выделите список и выберите команду Данные⇨Список⇨ Опубликовать список (Data⇨List⇨Publish List) из меню Excel. Запустится мастер Публикация списка на узле SharePoint (Publish List to SharePoint) (рис. 21.7, рис. 21.8 и рис. 21.9). Внесите всю необходимую информацию, и публикация списка будет завершена. (На момент написания этой книги компанией Microsoft предоставлялась ознакомительная версия сервера SharePoint.)

В Excel предоставляются простые механизмы публикации, просмотра, синхронизации и удаления списков. Просто выберите соответствующую команду меню. Разработчикам может захотеться выполнить эту операцию в процессе работы кода.

В следующих фрагментах кода показано, как программно публиковать, синхронизировать, просматривать и удалять списки.

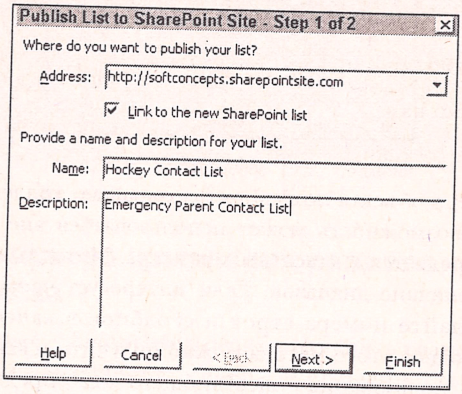


Рис. 21.7. Публикация списков на сервере SharePoint. Выбор сервера для публикации

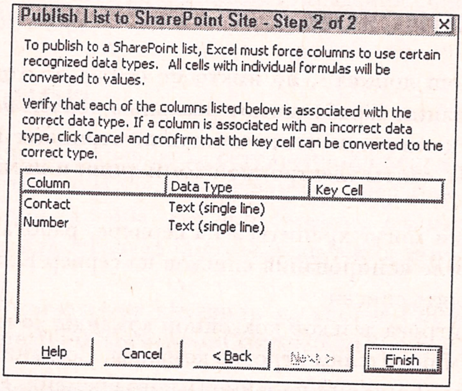


Рис. 21.8. Публикация списков на сервере SharePoint. Выбор публикуемых полей списка

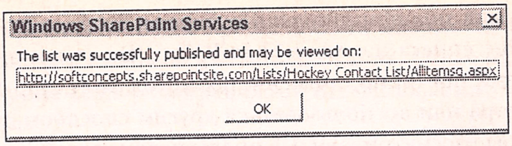


Рис. 21.9. Публикация списков на сервере SharePoint. Адрес опубликованного списка

Публикация списка

Предположим, существует список с контактной информацией. Вместо ручной публикации списка можно написать код, который выполнит автоматическую публикацию.

```
Public Sub PublishList()

    Dim Index As Integer
    ' жесткое кодирование индекса в целях тестирования
    Index = 1
    Dim List As ListObject
    Set List = GetList(Index)

    Dim Url As String
    Url = List.Publish(Array(
↳"http://softconcepts.sharepointsite.com",
↳"Hockey Moms Unite", "GLAHA Contact List"), True)
End Sub
```

(Обратите внимание: к моменту поступления книги в печать этот сайт SharePoint будет отключен. Адрес URL сервера SharePoint придется заменить на адрес действительного портала SharePoint.) Конкретный список можно получить по номеру или по индексу. В данном примере список выбирается по номеру. Этот же подход используется в следующих нескольких примерах. (Код подпрограммы `GetList` не будет указываться повторно.)

В этом примере извлекается конкретный объект `ListObject` и вызывается метод `ListObject.Publish`. В качестве параметров метода передается та же информация, что и в показанном ранее диалоговом окне. Функция `Array` используется для передачи в виде списка адреса URL, имени и описания. Второй аргумент указывает необходимость поддержки двунаправленной связи между сервером SharePoint и листом. Если параметр установлен в значение `True`, все изменения в опубликованном списке будут отражены в списке в процессе следующей синхронизации.

Внесение изменений в список

Теперь, когда список опубликован на сервере SharePoint, можно программно сохранять изменения списка на листе. Следующий код дает понять, насколько простым является этот процесс.

```
Public Sub SynchronizeList()
    ' жесткое кодирование индекса в целях тестирования
    Dim Index As Integer
    Index = 1

    Dim List As ListObject
    Set List = GetList(Index)

    Call List.UpdateChanges(XLListConflict.xlListConflictDialog)
End Sub
```

На рис. 21.10 показана модифицированная версия листа, а на рис. 21.11 — модифицированное содержимое списка на сайте SharePoint. (В этом коде используется жестко закодированный номер и повторно применяется метод `GetList`. Номер можно передавать в качестве параметра или воспользоваться другим способом динамического изменения номера; эти способы рассматривались в предыдущих главах.)

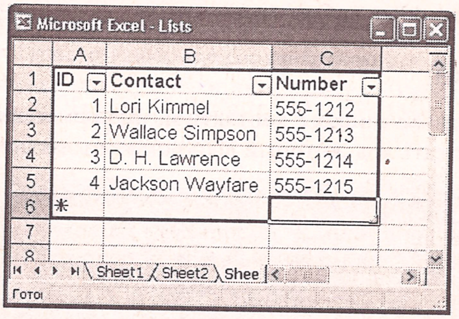


Рис. 21.10. Модифицированный список

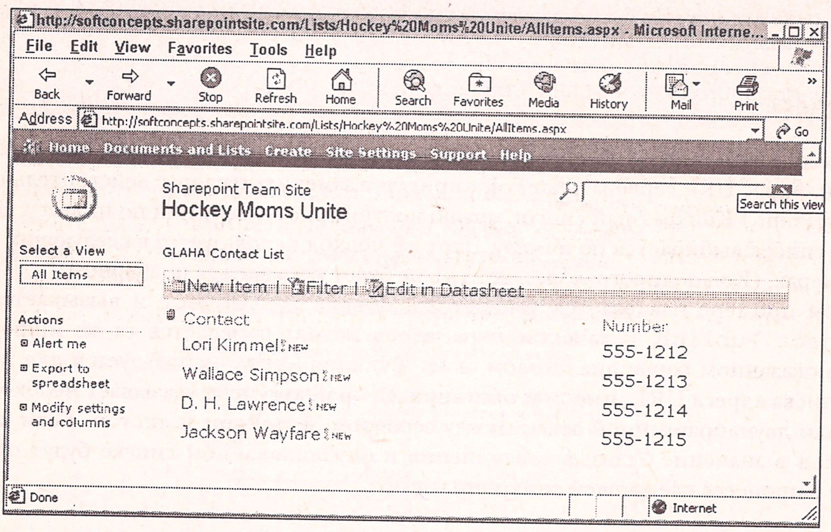


Рис. 21.11. Модифицированный список на сервере SharePoint

Просмотр списка на сервере SharePoint

Просмотр списка на сервере SharePoint ничем не отличается от просмотра любой другой Web-страницы, требующей авторизации. Для этого необходимо открыть интересующую страницу в обозревателе и авторизоваться в соответствии с требованием сервера SharePoint. В данном примере работа сайта обеспечивается партнером компании Microsoft и можно вручную открыть соответствующую страницу или выбрать команду **Данные⇨Список⇨Просмотреть список на сервере** (Data⇨List⇨View List on Server) в меню Excel.

Каждый сервер, скорее всего, будет предоставлять множество различных услуг и может требовать определенной платы за их использование. Обычно стоит самостоятельно разобраться с оплатой услуг сервера и правилами использования SharePoint.

Удаление списка

На определенном этапе содержимое списка может стать неактуальным. В таком случае список придется удалить с сервера. В данном примере список можно будет удалить после завершения хоккейного сезона. Для этого в Excel выберите команду меню Данные⇨Список⇨Удалить список (Data⇨List⇨Unlink List). Далее для фактического удаления списка можно воспользоваться ресурсами сайта SharePoint. Кроме этого, поддерживается программное удаление списков:

```
Public Sub UnlinkList()
    ' жесткое кодирование индекса в целях тестирования
    Dim Index As Integer
    Index = 1

    Dim List As ListObject
    Set List = GetList(Index)
    Call List.Publish
End Sub
```

Резюме

Microsoft Excel и серверы SharePoint предоставляют относительно простой способ публикации данных. В этой главе было показано, как создавать, фильтровать и сортировать списки. Кроме того, была рассмотрена врожденная связь между списками и диапазонами, а также показано, как использовать код обработки диапазонов для работы со списками. Во второй половине главы рассматривались публикация, синхронизация, просмотр и удаление списков. Все эти задачи могут выполняться как из Excel, так и из VBA.

22

Сводные таблицы

Сводные таблицы являются расширением таблиц с перекрестной табуляцией, которые используются для вывода статистической информации, а также для вывода сложных данных в табличном формате. В качестве примера перекрестной табуляции можно привести таблицу сотрудников организации, разбитую по полу и возрасту. Сводные таблицы являются более мощным инструментом и поддерживают вывод более двух переменных, поэтому они могут использоваться для вывода списка сотрудников в соответствии с полом, возрастом и уровнем алкоголя в крови (старая шутка статистиков). Сводные таблицы особенно полезны при выводе данных в формате, который не предоставляется готовым отчетом. Лучше всего при разработке приложения не только предоставлять пользователям отчеты, но и возможность извлекать данные для просмотра и обработки в Excel. Такой подход к разработке позволит сэкономить большой объем времени в будущем.

Исходные данные для сводной таблицы могут храниться на листе Excel, в текстовом файле, в базе данных Access или в любой другой внешней базе данных. В базе данных может храниться до 256 переменных (главное, чтобы пользователь смог интерпретировать результат). Сводные таблицы поддерживают множество типов стандартных расчетов, включая суммирование, подсчет количества и среднего значения. Кроме этого, сводные таблицы поддерживают подсчет частичных сумм и общих сумм.

Данные могут группироваться так же, как при использовании возможности сокрытия на листе Excel. Кроме этого, поддерживается сокрытие ненужных строк и столбцов. Также внутри таблицы можно определить расчеты. Сводные таблицы предоставляют значительную гибкость при изменении структуры данных и при добавлении или удалении переменных. В Excel 2003 поддерживается создание диаграмм, связанных с результатами в сводной таблице. При этом из диаграммы можно управлять структурой данных.

Сводные таблицы проектировались таким образом, чтобы сделать ручную генерацию и управление достаточно простыми. Для создания большого количества таблиц или дополнительной автоматизации операций можно воспользоваться объектной моделью Excel. В этой главе рассматриваются следующие объекты:

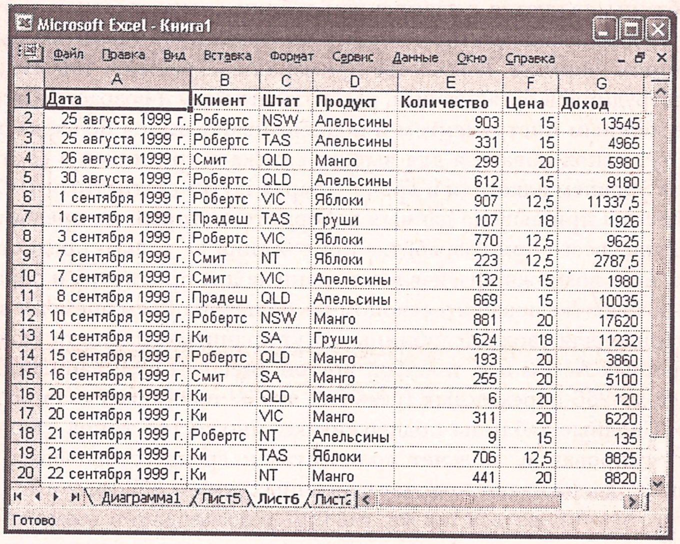
- `PivotTable`;
- `PivotCache`;
- `PivotField`;
- `PivotItem`;
- `PivotChart`.

Сводные таблицы являются наиболее развитой возможностью Excel. С каждой новой версией Excel сводные таблицы становились проще в использовании и предоставляли дополнительные возможности. Для большинства пользователей сохранение обратной совместимости между Excel 2002 и Excel 2003 является очень важным. Разница между этими версиями рассматривается далее.

Создание отчета для сводной таблицы

Сводная таблица принимает данные из листа или из внешнего источника, например, из базы данных Access. Данные из Excel должны быть структурированы в виде списка, как показано в начале главы 23 (хотя существует возможность использования данных из другой сводной таблицы или из нескольких консолидированных диапазонов). Столбцы списка являются полями, а строки — записями. В верхней строке определяются имена полей.

Рассмотрим список, содержащий входные данные (рис. 22.1).



	A	B	C	D	E	F	G
1	Дата	Клиент	Штат	Продукт	Количество	Цена	Доход
2	25 августа 1999 г.	Робертс	NSW	Апельсины	903	15	13545
3	25 августа 1999 г.	Робертс	TAS	Апельсины	331	15	4965
4	26 августа 1999 г.	Смит	QLD	Манго	299	20	5980
5	30 августа 1999 г.	Робертс	QLD	Апельсины	612	15	9180
6	1 сентября 1999 г.	Робертс	VIC	Яблоки	907	12,5	11337,5
7	1 сентября 1999 г.	Прадеш	TAS	Груши	107	18	1926
8	3 сентября 1999 г.	Робертс	VIC	Яблоки	770	12,5	9625
9	7 сентября 1999 г.	Смит	NT	Яблоки	223	12,5	2787,5
10	7 сентября 1999 г.	Смит	VIC	Апельсины	132	15	1980
11	8 сентября 1999 г.	Прадеш	QLD	Апельсины	669	15	10035
12	10 сентября 1999 г.	Робертс	NSW	Манго	881	20	17620
13	14 сентября 1999 г.	Ки	SA	Груши	624	18	11232
14	15 сентября 1999 г.	Робертс	QLD	Манго	193	20	3860
15	16 сентября 1999 г.	Смит	SA	Манго	255	20	5100
16	20 сентября 1999 г.	Ки	QLD	Манго	6	20	120
17	20 сентября 1999 г.	Ки	VIC	Манго	311	20	6220
18	21 сентября 1999 г.	Робертс	NT	Апельсины	9	15	135
19	21 сентября 1999 г.	Ки	TAS	Яблоки	706	12,5	8825
20	22 сентября 1999 г.	Ки	NT	Манго	441	20	8820

Рис. 22.1. Список с входными данными для сводной таблицы

Список содержит данные с августа 1999 года по декабрь 2001 года. Как обычно, в Excel желательно идентифицировать данные по имени, что намного упрощает обращение к ним из кода. В данном случае список называется `Database`. Excel автоматически обратится к этому имени при создании сводной таблицы.

Предположим, что необходимо подсчитать сумму поля Количество на протяжении всего временного периода по полям Клиент и Продукт. Разместив указатель в пределах списка данных, выберите команду Данные⇨Сводная таблица и отчет сводной диаграммы (Data⇨

Pivot Table and PivotChart Report). На третьем шаге мастера щелкните на кнопке Макет (Layout). Перетащите кнопку Клиент в область Строка (Row), кнопку Продукт (Product) в область Столбец (Column), а кнопку Количество в область Данные (Data) (рис. 22.2).

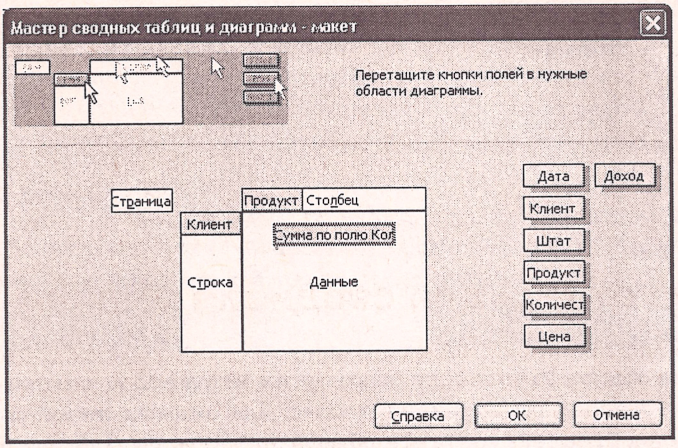


Рис. 22.2. Описание структуры сводной таблицы

Если выбрать размещение сводной таблицы на новом листе, то будет создан отчет сводной таблицы (рис. 22.3).

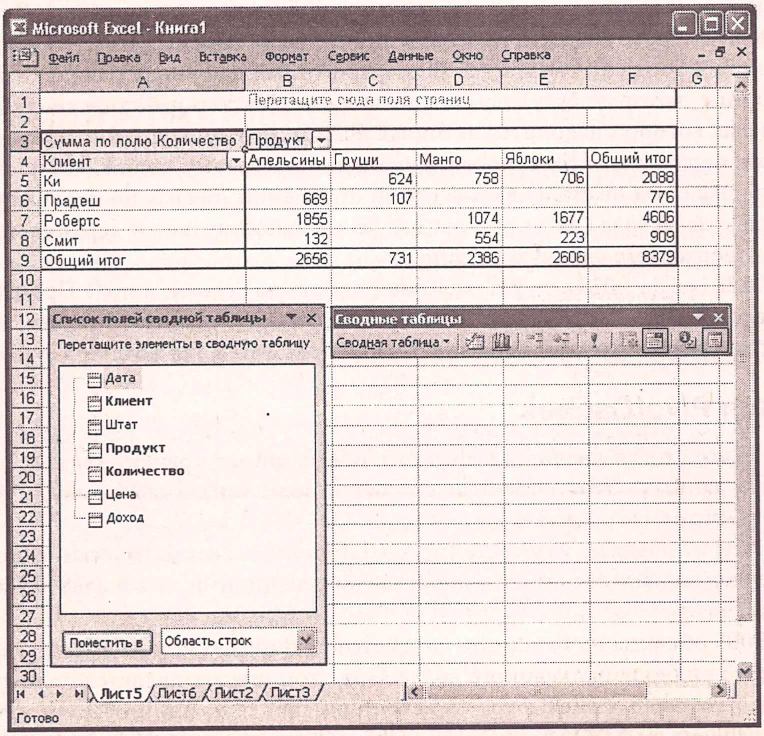


Рис. 22.3. Отчет сводной таблицы

Если при создании сводной таблицы в Excel 2003 включить механизм записи макросов, полученный код будет похож на показанный ниже фрагмент:

```
ActiveWorkbook.PivotCaches.Add _
    (SourceType:=xlDatabase, SourceData:="Database"). _
    CreatePivotTable TableDestination:="", _
    TableName:="СводнаяТаблица1", _
    DefaultVersion:=xlPivotTableVersion10

ActiveSheet.PivotTableWizard _
    TableDestination:=ActiveSheet.Cells(3, 1)
ActiveSheet.Cells(3, 1).Select

ActiveSheet.PivotTables("СводнаяТаблица1").AddFields _
    RowFields:="Клиент", ColumnFields:="Продукт"

ActiveSheet.PivotTables("СводнаяТаблица1"). _
    PivotFields("Количество"). _
    Orientation = xlDataField
```

Для создания объекта PivotCache используется метод Add коллекции PivotCaches. Коллекция PivotCaches рассматривается далее. Для создания пустой сводной таблицы применяется метод CreatePivotTable объекта PivotCache. Таблица размещается новом листе начиная с ячейки A1. Созданная таблица называется СводнаяТаблица1. Кроме этого, в данном случае используется новый параметр, доступный только в Excel 2002 и Excel 2003. Это параметр DefaultVersion, определяющий принятую по умолчанию версию сводной таблицы. В предыдущих версиях Excel этот параметр приводил к ошибке компиляции.

В следующей строке кода для перемещения сводной таблицы в ячейку A3 используется метод листа PivotTableWizard. В результате над таблицей появляется место для полей страницы, которые будут рассмотрены далее. Мастер работает со сводной таблицей, в пределах которой находится активная ячейка, поэтому перед использованием этого метода код не должен активизировать ячейки за пределами новой составной таблицы. Показанный выше код не сталкивается с этой проблемой, так как по умолчанию после завершения работы метода CreatePivotTable активной является верхняя левая ячейка диапазона, содержащего сводную таблицу.

Метод AddFields объявляет поле Клиент как поле строки, а поле Продукт как поле столбца. Наконец, поле Количество объявляется как поле данных. Это значит, что поле отображается в теле таблицы и по умолчанию используется для подсчета суммы.

Коллекция PivotCaches

Объект PivotCache является буфером и обеспечивает хранение и доступ к данным из источника данных. Этот объект выступает в роли канала связи между источником данных и объектом PivotTable.

Хотя в Excel 97 объекты PivotCache генерировались автоматически, явное создание объектов не поддерживалось и возможность управления ими, как в следующих версиях, не предоставлялась.

В Excel 2003 объекты PivotCache создаются с помощью метода Add коллекции PivotCaches (это видно из записанного кода). При этом предоставляется полный контроль над данными, которые извлекаются из источника при генерации объекта PivotCache. Вместе с интерфейсом ADO (ActiveX Data Objects) можно достичь значительной степени

контроля над внешними источниками данных (эта тема рассматривается ближе к концу главы). В главе 11 была продемонстрирована гибкость интерфейса ADO, а методики из этой главы позволяют создавать источники данных для сводных диаграмм.

Объекты PivotCache можно использовать для генерации нескольких сводных таблиц из одного источника данных. Это более эффективный подход, чем поддержка отдельного источника данных для каждой сводной таблицы.

При создании объекта PivotCache можно указывать любое количество сводных таблиц. Для этого используется метод CreatePivotTable объекта PivotCache.

Коллекция PivotTables

В Excel 2003 предоставляется другой метод создания сводной таблицы на основе объекта PivotCache. Это метод Add коллекции PivotTables.

```
Sub AddTable()
    Dim PC As PivotCache
    Dim PT As PivotTable

    Set PC = ActiveWorkbook.PivotCaches.Add(SourceType:=xlDatabase, _
             SourceData:="Database")

    Set PT = ActiveSheet.PivotTables.Add(PivotCache:=PC, _
             TableDestination:="")

End Sub
```

Данный метод не обладает дополнительными преимуществами по сравнению с методом CreatePivotTable. Просто еще одна возможность Excel.

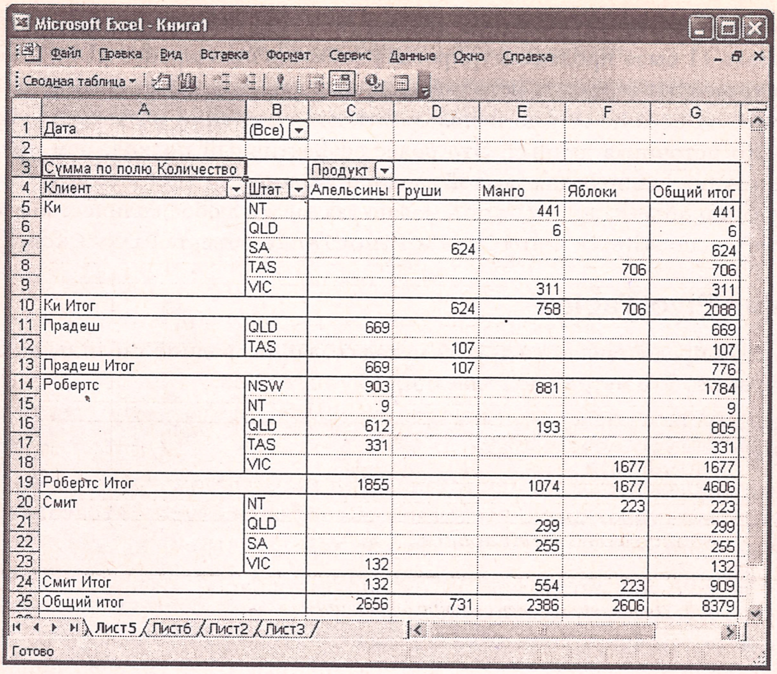
Коллекция PivotFields

Столбцы из источника данных называются полями. При загрузке полей в сводную таблицу они превращаются в объекты PivotField, принадлежащие коллекции PivotFields объекта PivotTable. В коллекции PivotFields хранятся все поля из источника данных, а также добавленные вычислимые поля (не только те поля, которые доступны в отчете сводной таблицы). Вычислимые поля рассматриваются далее в этой главе.

Для добавления объектов PivotFields в отчет можно воспользоваться двумя методиками. Можно применить метод AddFields объекта PivotTable или присвоить значение свойству Orientation объекта PivotField:

```
Sub AddFieldsToTable()

    With ActiveSheet.PivotTables(1)

        .AddFields RowFields:="State", AddToTable:=True
        .PivotFields("Date").Orientation = xlPageField

    End With
End Sub
```

При запуске этого кода вместе с примером сводной таблицы будет получен следующий результат (рис. 22.4).



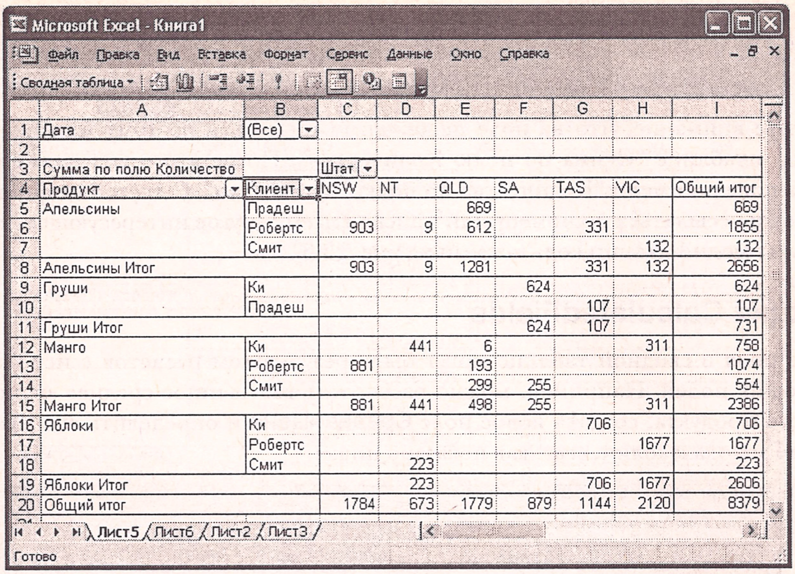
	A	B	C	D	E	F	G
1	Дата	(Все)					
2							
3	Сумма по полю Количество		Продукт				
4	Клиент	Штат	Апельсины	Груши	Манго	Яблоки	Общий итог
5	Ки	NT			441		441
6		QLD			6		6
7		SA		624			624
8		TAS				706	706
9		VIC			311		311
10	Ки Итог			624	758	706	2088
11	Прадеш	QLD	669				669
12		TAS		107			107
13	Прадеш Итог		669	107			776
14	Робертс	NSW	903		881		1784
15		NT	9				9
16		QLD	612		193		805
17		TAS	331				331
18		VIC				1677	1677
19	Робертс Итог		1855		1074	1677	4606
20	Смит	NT				223	223
21		QLD			299		299
22		SA			255		255
23		VIC	132				132
24	Смит Итог		132		554	223	909
25	Общий итог		2656	731	2386	2606	8379

Рис. 22.4. Добавление полей в сводную таблицу

Метод AddFields может добавлять несколько строк, столбцов и полей страниц. Если не установить параметр AddToTable в значение True (как показано в предыдущем примере), добавляемые поля будут замещать существующие поля. Но метод AddFields не поддерживает добавление или замещение полей данных. Следующий код переопределяет структуру полей существующей таблицы (не затрагивая поля данных).

```
Sub RedefinePivotTable()
    Dim PT As PivotTable

    Set PT = ActiveSheet.PivotTables(1)
    PT.AddFields RowFields:=Array("Product", "Customer"), _
        ColumnFields:="State", PageFields:="Date"

End Sub
```

Обратите внимание, что функцию массива можно использовать для добавления нескольких полей одновременно (рис. 22.5).



	A	B	C	D	E	F	G	H	I
1	Дата	(Все)							
2									
3	Сумма по полю Количество		Штат						
4	Продукт	Клиент	NSW	NT	QLD	SA	TAS	VIC	Общий итог
5	Апельсины	Прадеш			669				669
6		Робертс	903	9	612		331		1855
7		Смит						132	132
8	Апельсины Итог		903	9	1281		331	132	2656
9	Груши	Ки				624			624
10		Прадеш					107		107
11	Груши Итог					624	107		731
12	Манго	Ки		441	6			311	758
13		Робертс	881		193				1074
14		Смит			299	255			554
15	Манго Итог		881	441	498	255		311	2386
16	Яблоки	Ки					706		706
17		Робертс						1677	1677
18		Смит		223					223
19	Яблоки Итог			223			706	1677	2606
20	Общий итог		1784	673	1779	879	1144	2120	8379

Рис. 22.5. Добавление полей в сводную таблицу

Свойства `Orientation` и `Position` объекта `PivotField` позволяют реорганизовать структуру таблицы. Положение определяет порядок полей в пределах определенной части таблицы, начиная с поля слева. Если добавить следующий код в конец подпрограммы `RedefinePivotTable`, поле Клиент будет находиться слева от поля Продукт, как показано выше.

```
PT.PivotFields("Customer").Position = 1
```

Свойство `Function` объекта `PivotField` можно использовать для изменения механизма, с помощью которого поле данных учитывается в общей сумме. Свойство `NumberFormat` позволяет изменить представление чисел. Если вставить следующий код в конец подпрограммы `RedefinePivotTable`, в область данных будет добавлено поле Доход. По умолчанию оно будет добавлено после поля Количество. В следующих строках кода поле Количество перемещается на вторую позицию и меняется с "Сумма по полю Количество" на "Количество по полю Количество". В результате поле содержит количество транзакций по продаже.

```
Sub AddDataField()
    Dim PT As PivotTable

    Set PT = ActiveSheet.PivotTables(1)

    ' Добавить и отформатировать новое поле Data
    With PT.PivotFields("Доход")
        .Orientation = xlDataField
        .NumberFormat = "0"
    End With

    ' Отредактировать существующее поле Data
    With PT.DataFields("Сумма по полю Количество")
        .Position = 2
        .Function = xlCount
```

```
        .NumberFormat = "0"
    End With
End Sub
```

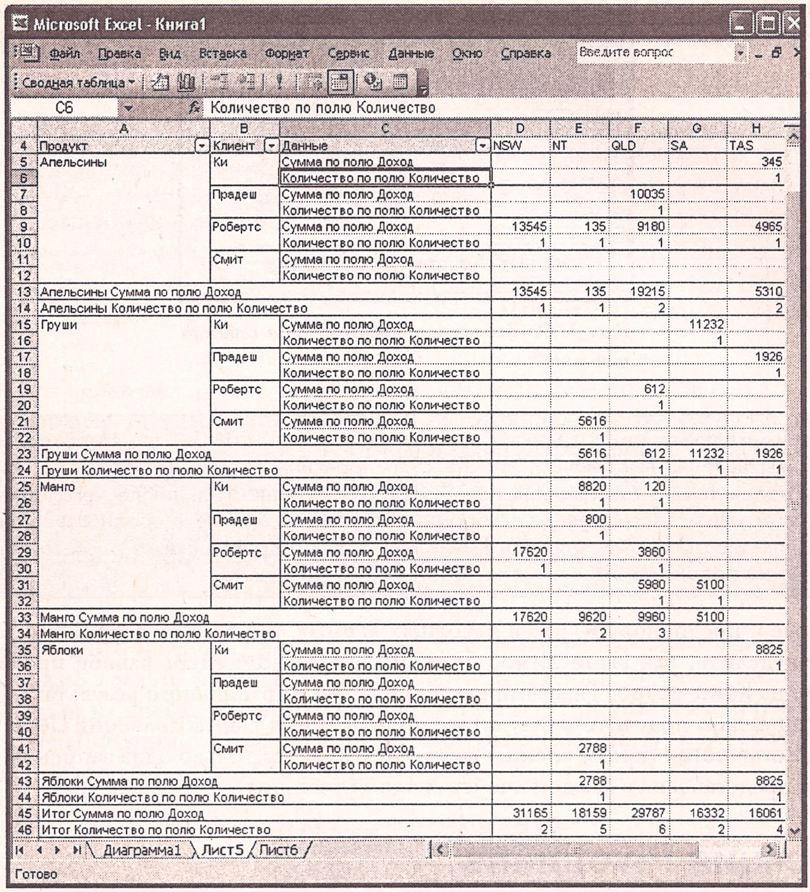
Обратите внимание, что на поле необходимо ссылаться по тому же имени, что используется в таблице "Сумма по полю Количество". После этого код должен будет ссылаться на поле по имени "Количество по полю Количество". С другой стороны, на поле данных можно ссылаться по номеру или присвоить ему другое интересующее имя.

Результат модификации кода показан на рис. 22.6.

Коллекция CalculatedFields

Новые поля в сводной таблице могут быть результатом расчетов с использованием существующих полей. Например, можно рассчитать взвешенное среднее значение цены для каждого продукта, создать новое поле СредняяЦена и определить его значение как отношение дохода к количеству, как показано ниже:

```
Sub CalculateAveragePrice()
    Dim PT As PivotTable

    ' Добавить новый лист и сводную таблицу
    Worksheets.Add
    Set PT = ActiveWorkbook.PivotCaches(1).CreatePivotTable( _
            TableDestination:=ActiveCell, TableName:="СредняяЦена")
    With PT

        ' Удалить существующее поле СредняяЦена
        On Error Resume Next
        .PivotFields("СредняяЦена").Delete
        On Error GoTo 0

        ' Создать новое поле СредняяЦена
        .CalculatedFields.Add Name:="СредняяЦена", _
            Formula:="=Доход/Количество"

        ' Добавить поля Строка и Столбец
        .AddFields RowFields:="Клиент", ColumnFields:="Продукт"

        ' Добавить поле СредняяЦена типа Data
         With .PivotFields("СредняяЦена")
             .Orientation = xlDataField
             .NumberFormat = "0.00"
         End With
         ' Удалить итоговые суммы
         .ColumnGrand = False
         .RowGrand = False

    End With

End Sub
```



	A	B	C	D	E	F	G	H
4	Продукт	Клиент	Данные	NSW	NT	QLD	SA	TAS
5	Апельсины	Ки	Сумма по полю Доход					345
6			Количество по полю Количество					1
7		Прадеш	Сумма по полю Доход			10035		
8			Количество по полю Количество			1		
9		Робертс	Сумма по полю Доход	13545	135	9180		4965
10			Количество по полю Количество	1	1	1		1
11		Смит	Сумма по полю Доход					
12			Количество по полю Количество					
13	Апельсины Сумма по полю Доход			13545	135	19215		5310
14	Апельсины Количество по полю Количество			1	1	2		2
15	Груши	Ки	Сумма по полю Доход				11232	
16			Количество по полю Количество				1	
17		Прадеш	Сумма по полю Доход					1926
18			Количество по полю Количество					1
19		Робертс	Сумма по полю Доход			612		
20			Количество по полю Количество			1		
21		Смит	Сумма по полю Доход		5616			
22			Количество по полю Количество		1			
23	Груши Сумма по полю Доход				5616	612	11232	1926
24	Груши Количество по полю Количество				1	1	1	1
25	Манго	Ки	Сумма по полю Доход		8820	120		
26			Количество по полю Количество		1	1		
27		Прадеш	Сумма по полю Доход		800			
28			Количество по полю Количество		1			
29		Робертс	Сумма по полю Доход	17620		3860		
30			Количество по полю Количество	1		1		
31		Смит	Сумма по полю Доход			5980	5100	
32			Количество по полю Количество			1	1	
33	Манго Сумма по полю Доход			17620	9620	9960	5100	
34	Манго Количество по полю Количество			1	2	3	1	
35	Яблоки	Ки	Сумма по полю Доход					8825
36			Количество по полю Количество					1
37		Прадеш	Сумма по полю Доход					
38			Количество по полю Количество					
39		Робертс	Сумма по полю Доход					
40			Количество по полю Количество					
41		Смит	Сумма по полю Доход		2788			
42			Количество по полю Количество		1			
43	Яблоки Сумма по полю Доход				2788			8825
44	Яблоки Количество по полю Количество				1			1
45	Итог Сумма по полю Доход			31165	18159	29787	16332	16061
46	Итог Количество по полю Количество			2	5	6	2	4

Рис. 22.6. Добавление нового поля

Подпрограмма CalculateAveragePrice добавляет новый лист и использует метод CreatePivotTable созданного ранее объекта PivotCache для создания новой сводной таблицы на новом листе. Для обеспечения повторного запуска код удаляет существующий объект PivotField, который называется СредняяЦена. Операторы On Error обеспечивают работоспособность кода даже в случае отсутствия поля СредняяЦена.

Для доступа к коллекции CalculatedFields используется метод CalculatedFields объекта PivotTable. Метод Add коллекции CalculatedFields позволяет добавить новое поле. Обратите внимание, что несмотря на кажущееся введение поля в сводную таблицу, на самом деле поле добавляется в PivotCache. Добавленное поле доступно и в первой сводной таблице. Удаление новой сводной таблицы не приведет к удалению поля СредняяЦена из объекта PivotCache. После добавления нового поля его можно обрабатывать, как и любое другое поле из коллекции PivotFields. В последней строке кода удаляются поля суммы, создаваемые по умолчанию.

Результат показан ниже (рис. 22.7). Так как цены в источнике данных не меняются, не вызывает удивления отсутствие изменений средневзвешенных цен для каждого продукта в сводной таблице.

	A	B	C	D	E	F
1	Клиент	Апельсины	Груши	Манго	Яблоки	
2	Ки	15,00	18,00	20,00	12,50	
3	Прадеш	15,00	18,00	20,00	12,50	
4	Робертс	15,00	18,00	20,00	12,50	
5	Смит	15,00	18,00	20,00	12,50	
6						
7						

Рис. 22.7. Добавление нового поля в сводную таблицу

> Будьте внимательны при создании коллекции `CalculatedFields`. Необходимо помнить, что вычисления выполняются после суммирования данных из источника. В данном примере подсчитываются значения полей Доход и Количество, после чего одна сумма делится на другую. Такое решение хорошо подходит для расчета средневзвешенной цены и/или для простого сложения или вычитания. Но при другой схеме расчетов этот подход может не подойти.

Например, предположим, что в исходных данных отсутствует поле Доход. Это поле можно определить, как вычислимое поле (`CalculatedField`), равное произведению поля Цена на Количество. Такое определение не даст правильного результата. Значение поля Доход нельзя получить в результате произведения суммы значений Цена на сумму значений Количество (кроме специального случая, когда только одна запись из источника данных представлена в каждой ячейке сводной таблицы).

Коллекция PivotItems

С каждым объектом `PivotField` связана коллекция `PivotItems`. Для доступа к коллекции `PivotItems` можно воспользоваться методом `PivotItems` объекта `PivotField`. Тот факт, что это не свойство, а метод, вызывает некоторое удивление. Это отличает метод `PivotItems` от свойств `HiddenItems` и `VisibleItems` объекта `PivotField`, которые возвращают подмножества коллекции `PivotItems`.

В коллекции `PivotItems` содержатся уникальные значения поля. Например, поле Продукт в исходных данных может содержать четыре уникальных значения — “Яблоки”, “Манго”, “Апельсины” и “Груши”. Эти значения составляют коллекцию `PivotItems` для этого поля.

Группирование

Excel поддерживает любой способ группирования элементов поля. Например, элементы NSW, QLD и VIC могут быть сгруппированы в ВосточныеШтаты. Эта возможность требуется, когда в поле хранится большое количество элементов. Кроме этого, поддерживается группирование дат с использованием предопределенной структуры группы, включая год, квартал и месяц.

При переносе поля Дата из источника данных в сводную таблицу в виде поля строки в таблице окажется около 400 строк, так как в источнике хранится множество уникальных дат (рис. 22.8).

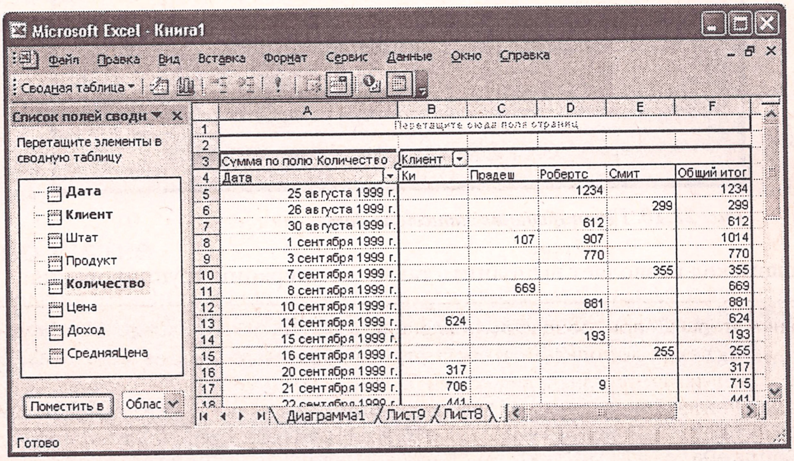


Рис. 22.8. Группирование значений даты

Элементы Дата можно сгруппировать в более осмысленные итоговые характеристики. Это можно сделать вручную, выбрав ячейку в сводной таблице, щелкнув правой кнопкой мыши на ячейке и выбрав команду Группа и структура⇨Группировать (Group and Show Detail⇨Group). Откроется диалоговое окно (рис. 22.9), в котором можно выбрать пункт Месяц (Month) и Год (Year).

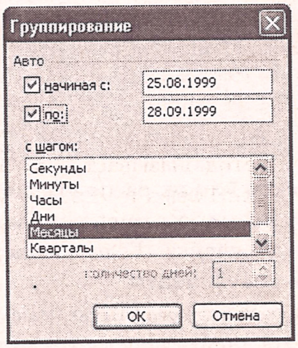
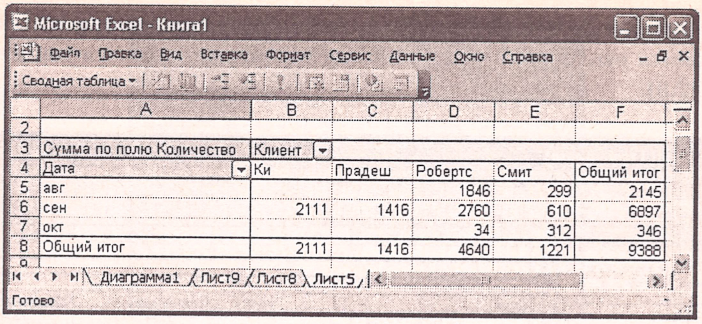


Рис. 22.9. Ручное группирование дат

После щелчка на кнопке ОК таблица будет выглядеть следующим образом (рис. 22.10).



	A	B	C	D	E	F
2						
3	Сумма по полю Количество	Клиент				
4	Дата	Ки	Традеш	Робертс	Смит	Общий итог
5	авг			1846	299	2145
6	сен	2111	1416	2760	610	6897
7	окт			34	312	346
8	Общий итог	2111	1416	4640	1221	9388

Рис. 22.10. Группирование записей по дате

Следующий код позволяет выполнить такую же операцию группирования:

```
Sub GroupDates()
    Dim PT As PivotTable
    Dim Rng As Range

    ' Создать новую сводную таблицу в ячейке A1 активного листа
    Set PT = ActiveSheet.PivotTableWizard(SourceType:=xlDatabase, _
        SourceData:=ThisWorkbook.Names("Database").RefersToRange, _
        TableDestination:=Range("A1"))

    With PT
        ' Добавить данные
        .AddFields RowFields:="Дата", ColumnFields:="Штат"
        .PivotFields("Количество").Orientation = xlDataField

        ' Найти метку Дата
        Set Rng = .PivotFields("Дата").LabelRange

        ' Сгруппировать все даты по месяцам и годам
        Rng.Group Start:=True, End:=True, _
        Periods:=Array(False, False, False, False, True, False, True)

    End With
End Sub
```

При этом группируются объекты Range, расположенные под меткой поля или меткой элементов поля. Можно выбрать как метку с именем поля, так и метку с именем элемента. Главное — выбрать единственную ячейку. Если выбрать несколько имен элементов, в группу будут добавлены только эти элементы.

Подпрограмма GroupDates создает объектную переменную Rng, которая ссылается на ячейку с меткой поля **Дата**. Эта ячейка передается методу Group вместе с параметрами, относящимися к датам. Параметры Start и End определяют начальную и конечную даты. Если эти параметры установлены в значение True, в группу включаются все даты. Массив в параметре Periods соответствует пользовательскому вводу в диалоговом окне **Группирование** (Grouping). Этот массив описывает выбор месяца и года.

Следующий код позволяет разгруппировать даты:

```
Sub UnGroupDates()
    Dim Rng As Range

    Set Rng = ActiveSheet.PivotTables(1).
```

```
↳PivotFields("Дата").LabelRange
    Rng.Ungroup
End Sub
```

Повторная группировка выполняется с помощью следующего кода:

```
Sub ReGroupDates()
    Dim Rng As Range

    Set Rng = ActiveSheet.PivotTables(1).PivotFields("Date").LabelRange
    Rng.Group Start:=True, End:=True, _
      Periods:=Array(False, False, False, False, True, False, True)

End Sub
```

Свойство Visible

Для сокрытия элемента можно установить его свойство `Visible` в значение `False`. Предположим, что приходится работать со сгруппированными датами из предыдущего примера и необходим только январь 2000 года и январь 2001 года (рис. 22.11).

2						
3	Sum of NumberSold		State			
4	Years	Date	NT	TAS	VIC	Grand Total
5	2000	Jan	597	432	1024	2053
6	2001	Jan	73	1230	89	1392
7	Grand Total		670	1662	1113	3445

Рис. 22.11. Сокрытие некоторых элементов поля

Для этого можно воспользоваться следующим кодом:

```
Sub CompareMonths()
    Dim PT As PivotTable
    Dim PI As PivotItem
    Dim sMonth As String

    sMonth = "Янв"
    Set PT = ActiveSheet.PivotTables(1)

    For Each PI In PT.PivotFields("Годы").PivotItems
        If PI.Name <> "2000" And PI.Name <> "2001" Then
            PI.Visible = False
        End If
    Next PI

    PT.PivotFields("Дата").PivotItems(sMonth).Visible = True
    For Each PI In PT.PivotFields("Дата").PivotItems
        If PI.Name <> sMonth Then PI.Visible = False
    Next PI
End Sub
```

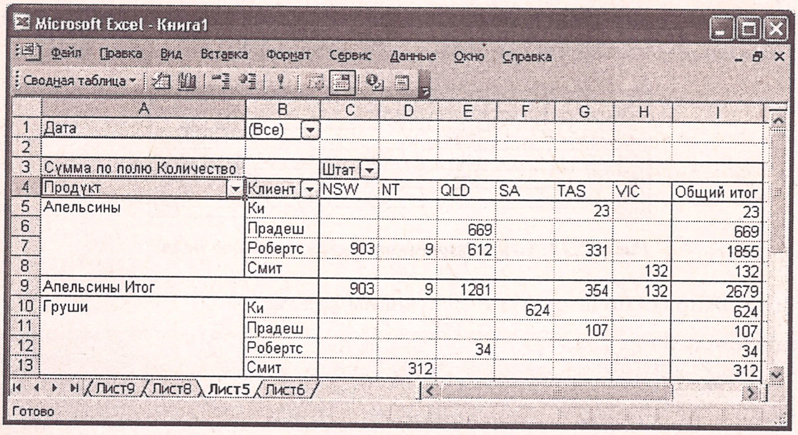
Подпрограмма `CompareMonths` циклически перебирает все элементы в полях Годы и Дата, устанавливая свойство `Visible` в значение `False`, если текущий элемент должен быть сокрыт. Этот код может использоваться повторно. Для сравнения других месяцев можно присваивать новые значения переменной `sMonth`. Обратите внимание, что интересующий месяц должен быть видимым до начала обработки элементов поля Дата. Это необходимо для обеспечения видимости интересующего месяца и защиты от одновременного сокрытия всех элементов. Если ни один из элементов не будет видимым, возникнет ошибка времени выполнения.

Коллекция CalculatedItems

Для добавления вычислимых элементов в поле воспользуйтесь методом Add коллекции CalculatedItems. Предположим, что необходимо добавить новый продукт — "Дыня". Предполагается, что дынь будет продаваться на 50% больше, чем манго. Для включения этой информации в таблицу можно воспользоваться следующим кодом:

```
Sub AddCalculatedItem()

    With ActiveSheet.PivotTables(1).PivotFields("Product")
        .CalculatedItems.Add Name:="Дыня", Formula:="=Манго*1.5"
    End With

End Sub
```

Результат работы этого кода показан на рис. 22.12.



	A	B	C	D	E	F	G	H	I
1	Дата	(Все)							
2									
3	Сумма по полю Количество		Штат						
4	Продукт	Клиент	NSW	NT	QLD	SA	TAS	VIC	Общий итог
5	Апельсины	Ки					23		23
6		Прадеш			669				669
7		Робертс	903	9	612		331		1855
8		Смит						132	132
9	Апельсины Итог		903	9	1281		354	132	2679
10	Груши	Ки				624			624
11		Прадеш					107		107
12		Робертс			34				34
13		Смит		312					312

Рис. 22.12. Добавление в поле вычислимого элемента

Для удаления объекта CalculatedItem его нужно удалить из коллекции CalculatedItems или из коллекции PivotItems объекта PivotField:

```
Sub DeleteCalculatedItem()
    With ActiveSheet.PivotTables(1).PivotFields("Продукт")
        .PivotItems("Дыня").Delete
    End With
End Sub
```

Сводные диаграммы

Сводные диаграммы впервые появились в Excel 2000. Они следуют всем правилам, определенным для объектов Chart, но связываются с объектами PivotTable. При изменении структуры сводной диаграммы Excel автоматически меняет структуру связанной сводной таблицы. Соответственно, при изменении структуры сводной таблицы автоматически меняется структура связанной сводной диаграммы.

Следующий код создает новую сводную таблицу на основе того же объекта PivotChart, который использовался для создания сводной таблицы на листе Sheet1. После этого на основе сводной таблицы создается сводная диаграмма (объект PivotChart).

```
Sub CreatePivotChart()
    Dim PC As PivotCache
    Dim PT As PivotTable
    Dim Cht As Chart

    ' Получить ссылку на существующий объект PivotCache
    Set PC = Worksheets("Sheet1").PivotTables(1).PivotCache

    ' Добавить новый лист
    Worksheets.Add Before:=Worksheets(1)

    ' Создать сводную таблицу на основе существующего
    ' объекта PivotCache
    Set PT = PC.CreatePivotTable(TableDestination:=ActiveCell)

    ' Добавить лист диаграммы на основе активной сводной таблицы
    Set Cht = Charts.Add(Before:=Worksheets(1))

    ' Отформатировать сводную диаграмму
    With Cht.PivotLayout.PivotTable

        .PivotFields("Клиент").Orientation = xlRowField
        .PivotFields("Штат").Orientation = xlColumnField
        .PivotFields("Количество").Orientation = xlDataField
        '.AddDataField .PivotFields("Количество"), "Sum of NumberSold", _
xlSum

    End With

End Sub
```

В результате этого кода получена диаграмма, показанная на рис. 22.13.

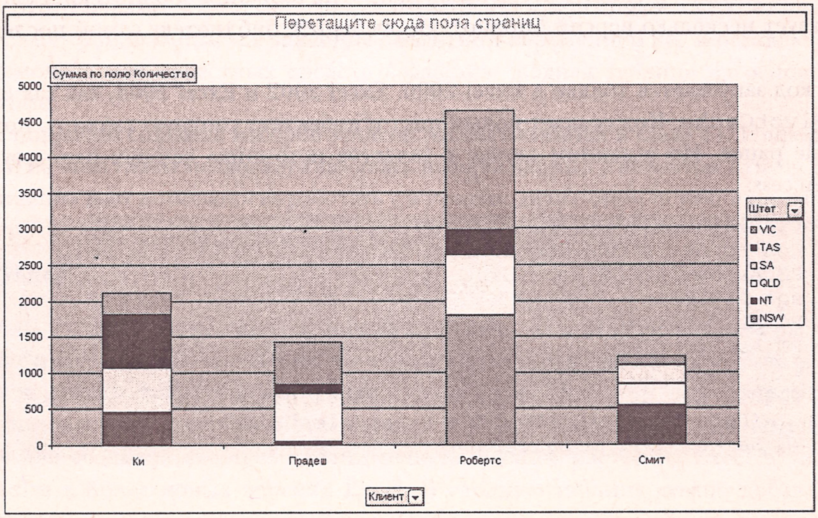


Рис. 22.13. Сводная диаграмма

После создания сводной таблицы на новом листе код добавляет новую диаграмму. Для этого используется метод `Add` коллекции `Charts`. Важно, чтобы в момент вызова метода `Add` активная ячейка находилась в пределах сводной таблицы. При этом новая диаграмма автоматически будет связана со сводной таблицей.

После создания сводной диаграммы ее можно связать со сводной таблицей с помощью объекта `PivotLayout` (этот объект связан с диаграммой). После этого достаточно настроить необходимые поля сводной таблицы, воспользовавшись уже известными приемами.

Последняя строка перед оператором `End With` закомментирована. Это альтернативный способ добавления поля данных. Эту строку можно использовать вместе предыдущей, но метод `AddDataField` объекта `PivotTable` доступен только в Excel 2002 и Excel 2003.

Последующие манипуляции данными на диаграмме должны выполняться только через соответствующую сводную таблицу. Для форматирования изменений на диаграмме можно воспользоваться свойствами и методами объекта `Chart`.

Внешние источники данных

Excel поддерживает хранение ограниченного объема данных. Кроме этого, Excel плохо справляется с хранением нескольких связанных таблиц с данными. Таким образом, приходится хранить данные во внешней базе данных и извлекать их по мере необходимости. Самым мощным решением является использование интерфейса ADO (ActiveX Data Objects). Более подробно интерфейс ADO рассматривался в главе 11.

В следующем примере показано, как подключаться к базе данных Access в файле `SalesDB.mdb`, в которой хранятся данные, используемые ранее, но также могут храниться намного более полные и сложные данные. Для запуска кода необходимо создать ссылку на библиотеку ADO. Для этого переключитесь в окно редактора VBE и выберите пункт Tools⇨References (Сервис⇨Ссылки). В открывшемся списке выберите "Microsoft ActiveX Data Objects" и установите флажок напротив названия библиотеки. Если в списке присутствует несколько версий библиотеки, выберите библиотеку самой последней версии.

Этот код запускается только в Excel 2000, Excel 2002 и Excel 2003, так как он создает объект `PivotCache`. Код создает новый лист и добавляет сводную таблицу, похожую на созданные ранее. Но в данном случае в качестве источника данных используется база данных Access:

```
Sub PivotTableDataViaADO()
    Dim Con As New ADODB.Connection
    Dim RS As New ADODB.Recordset
    Dim sql As String
    Dim PC As PivotCache
    Dim PT As PivotTable

    Con.Open "Provider=Microsoft.Jet.OLEDB.4.0;" & _
            "Data Source=C:\My Documents\SalesDB.mdb;"
    sql = "Select * From SalesData"
```

```
    ' Открыть список записей
    Set RS = New ADODB.Recordset
    Set RS.ActiveConnection = Con
    RS.Open sql

    ' Создать кэш сводной таблицы
    Set PC = ActiveWorkbook.PivotCaches.Add(SourceType:=xlExternal)
    Set PC.Recordset = RS

    ' Создать сводную таблицу
    Worksheets.Add Before:=Sheets(1)
    Set PT = ActiveSheet.PivotTables.Add(PivotCache:=PC, _
            TableDestination:=Range("A1"))

    With PT
        .NullString = "0"
        .SmallGrid = False
        .AddFields RowFields:="Штат", ColumnFields:="Продукт"
        .PivotFields("Количество").Orientation = xlDataField
    End With
End Sub
```

Сначала создается объект Connection, устанавливающий подключение к базе данных Access. Для подключения используется метод Open объекта ADO Connection. После этого определяется оператор на языке SQL, в котором запрашиваются все записи из таблицы SalesData в базе данных Access. Таблица практически идентична той, которая используется в Excel (в таблице содержатся те же поля и данные). Дополнительная информация о языке SQL и терминологии ADO приводится в главе 11.

После этого ссылка на объект Recordset присваивается объектной переменной RS. Свойству ActiveConnection объекта RS присваивается ссылка на объект Connection. Далее метод Open заполняет набор записей данными из таблицы SalesData в соответствии с инструкциями в операторе SQL.

После этого открывается объект PivotCache и объявляется внешний источник данных. Для этого параметр SourceType устанавливается в значение xlExternal, а свойству Recordset присваивается значение объектной переменной RS. В оставшейся части кода с помощью уже известных средств создается сводная таблица на основе объекта PivotCache.

Более подробно создание наборов записей рассматривается в главе 11. Информация из главы 11 и из этой главы позволяет связывать наборы данных с объектом PivotCache. Такая комбинация дает возможность работать с широким диапазоном источников данных.

Резюме

Сводные таблицы позволяют выводить сложные данные. В этой главе рассматривались различные способы программного создания сводных таблиц на основе различных источников данных, например на основе списков Excel. При этом указывались особенности кода, позволяющие сохранять совместимость между различными версиями Excel.

Также для создания сводных таблиц использовались мастер из Excel 97 и объекты PivotCache в более новых версиях Excel. В сводную таблицу можно добавлять поля в виде строки, столбца или поля данных. Поля могут содержать результат расчетов на основе других полей или элементов других полей. Элементы можно группировать. На-

пример, возможность группирования может использоваться для комбинирования дат по годам и месяцам. Кроме этого, элементы полей можно скрывать.

Со сводной таблицей можно связать сводную диаграмму. После создания такой связи изменения в одном компоненте будут отражаться на содержимом другого компонента. Для связи сводной таблицы и сводной диаграммы используется объект `PivotLayout`.

Интерфейс ADO позволяет связывать сводные таблицы с внешними источниками данных.

23

Списки с фильтрами

В этой главе рассматривается создание кода VBA для управления данными и фильтрации информации в списках. Также описываются средства Данные⇨Форма (Data⇨Form), Автофильтр (AutoFilter) и Расширенный фильтр (Advanced Filter). Не останутся без внимания и вопросы создания списков с фильтрацией, генерации служебного кода с помощью механизма записи макросов, а также использования элементов управления для вывода содержимого списков. Кроме того, демонстрируется, как обойти возможные проблемы, связанные с интернационализацией приложений.

Структурирование данных

Перед использованием инструментов управления списками Excel данные должны быть настроены особым образом, а именно: структурированы в виде таблицы из базы данных. В верхней части каждого столбца должен находиться заголовок с именем поля, а данные должны быть записаны в строках, соответствующих записям в базе данных. На рис. 23.1 показан список с информацией о продажах фруктов.

> Не стоит рассматривать Excel в качестве полноценного приложения для работы с базами данных. Excel поддерживает наборы данных ограниченного объема и не в состоянии обрабатывать несколько взаимосвязанных таблиц. Но Excel может работать с другими приложениями баз данных и предоставлять необходимые данные с помощью таких мощных инструментов, как Данные⇨Форма (Data⇨Form), Автофильтр (AutoFilter), Расширенный фильтр (Advanced Filter), Промежуточные.Итоги (SubTotal) и Сводная таблица (PivotTable). Эти инструменты позволяют анализировать, модифицировать и выводить данные в удобной форме. Примеры использования этих инструментов приводились в главе 22.

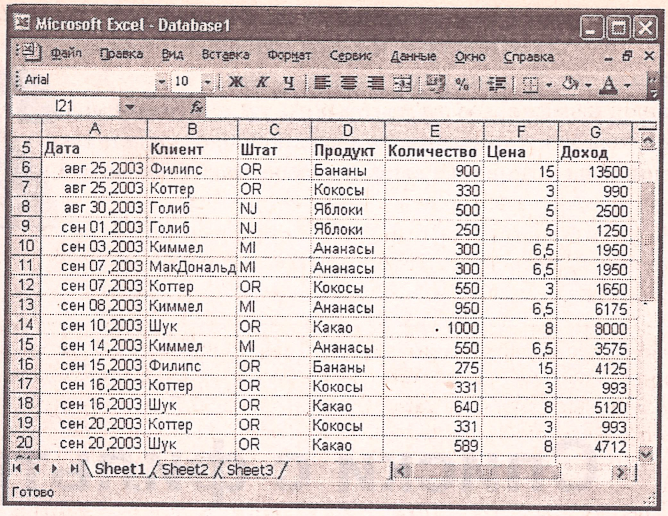


Рис. 23.1. Список с данными

Команда Форма

Excel предоставляет встроенное диалоговое окно, которое может использоваться для просмотра, поиска и редактирования данных в списках. При выделении единственной ячейки списка или же всего списка можно выбрать команду меню Данные⇨Форма (Data⇨Form) и откроется следующее диалоговое окно (рис. 23.2).

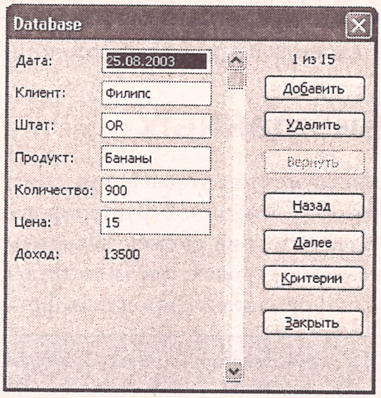


Рис. 23.2. Диалоговое окно для просмотра, поиска и редактирования данных

Механизм записи макросов позволяет получить следующий код:

```
Sub Macro1()
'
' Macro1 Macro
' Макрос записан 11/16/2003 Полом Киммелом
'
```

```
    Range("B7").Select
    ActiveSheet.ShowDataForm
End Sub
```

Если список начинается в диапазоне A5:B6 и в процессе записи макроса выделяется первая ячейка и открывается диалоговое окно Данные⇨Форма (Data⇨Form), записанный макрос будет работать. Если список начинается за пределами диапазона A5:B6 и в процессе записи макроса выделяется верхняя левая ячейка и выводится диалоговое окно Данные⇨Форма (Data⇨Form), при запуске записанного макроса будет выдаваться сообщение об ошибке. Для решения этой проблемы интересующему списку можно присвоить имя Database.

> Если приложение должно работать не только в региональных стандартах США и диалоговое окно Данные⇨Форма (Data⇨Form) выводится с помощью метода ShowDataForm, то возможность Данные⇨Форма (Data⇨Form) может быть связана с рядом опасностей. При вызове диалогового окна Данные⇨Форма (Data⇨Form) из VBA даты и числа выводятся только в соответствии с региональными стандартами США. С другой стороны, введенные пользователем числа и даты интерпретируются в соответствии с текущими региональными параметрами Windows. Таким образом, при вводе даты в соответствии с региональными стандартами Великобритании (дд/мм/гггг) в диалоговом окне Данные⇨Форма (Data⇨Form) дата будет повреждена. Дополнительная информация приводится в главе 17.

Автофильтр

Возможность Автофильтр (AutoFilter) обеспечивает простой способ выбора данных из списка. Для активизации автофильтра выберите пункт Данные⇨Фильтр⇨Автофильтр (Data⇨Filter⇨AutoFilter). Возле имени каждого поля появится раскрывающийся список. Если необходимо найти соответствие для конкретного поля, например Клиент, щелкните на раскрывающемся списке и выберите необходимое значение (рис. 23.3).

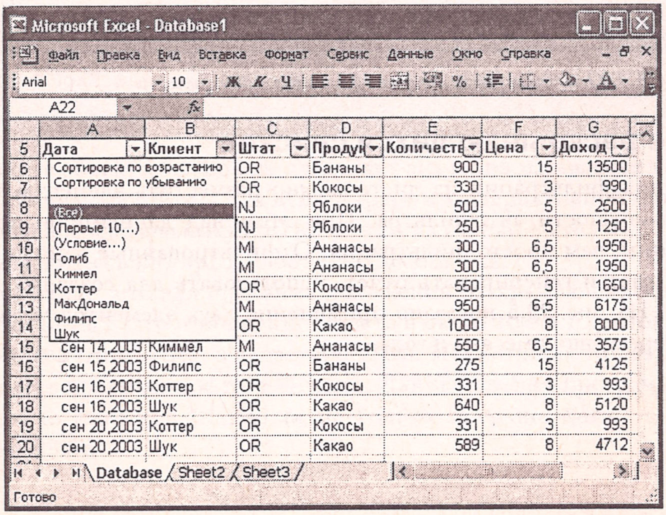


Рис. 23.3. Использование возможности Автофильтр

Пользовательский Автофильтр

Если необходимо получить более сложный фильтр, например диапазон дат, придется немного поработать. На экране (рис. 23.3) показано, как вручную фильтровать данные для отображения определенного месяца. Щелкните на раскрывающемся списке возле поля Дата и выберите Условие (Custom). Заполните поля в открывшемся диалоговом окне (рис. 23.4).

> Формат дат, применяемый в диалоговом окне Пользовательский автофильтр (Custom AutoFilter), зависит от региональных параметров. При работе с региональными параметрами Великобритании можно использовать формат дд/мм/гг. При работе с региональными параметрами США можно применять формат мм/дд/гг. Кроме этого, распознаются и другие форматы.

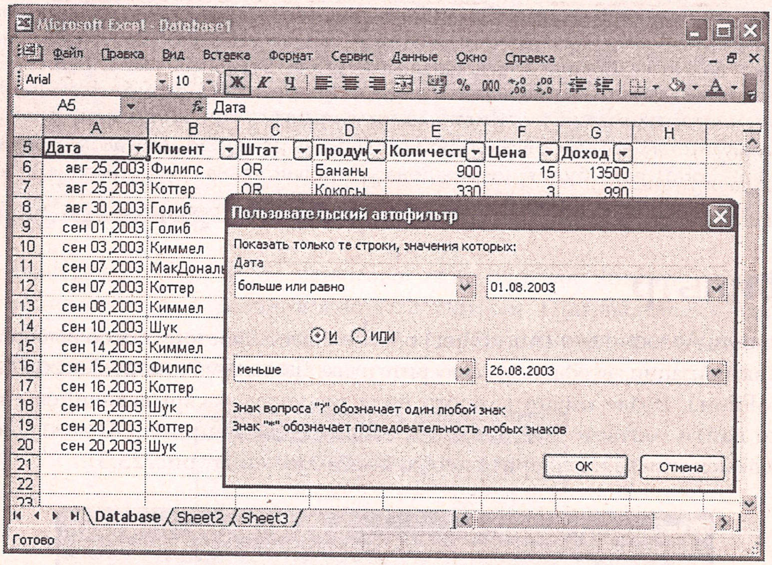


Рис. 23.4. Описание расширенного фильтра Автофильтр

Добавление раскрывающихся списков

Для упрощения фильтрации на листе можно разместить элементы управления, использующие возможности автофильтра. При этом над данными выполняются более сложные операции, чем просто фильтрация. Отфильтрованные данные можно копировать на другой лист и генерировать отчет, использовать для создания диаграммы или просто удалить. На рис. 23.5 показано применение двух элементов управления ActiveX для выбора интересующих месяца и года.

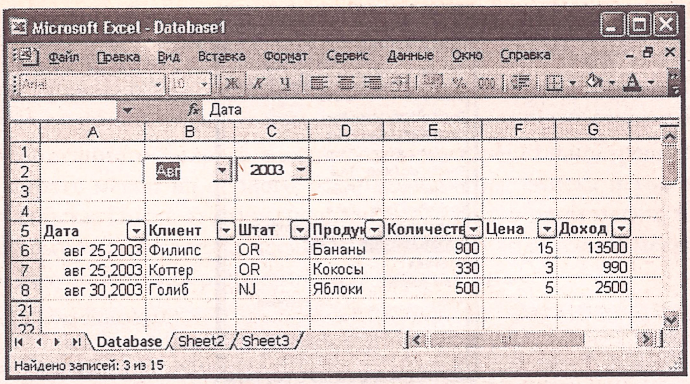


Рис. 23.5. Использование дополнительных элементов управления

Раскрывающиеся списки по умолчанию называются ComboBox1 и ComboBox2. Для внесения значений из списка в элементы управления можно присвоить свойству ListFillRange объекта ComboBox значение "=Sheet2!H1:H12". С другой стороны, можно воспользоваться процедурой обработки события Workbook_Open в модуле ThisWorkbook. Эта процедура вносит значения в элементы управления при открытии книги:

```
Private Sub Workbook_Open()
    Dim Months() As Variant
    Dim Years() As Variant
    Dim I As Integer

    Months = Array("Янв", "Фев", "Мар", "Апр", _
        "Май", "Июн", "Июл", "Авг", "Сен", _
        "Окт", "Ноя", "Дек")

    Years = Array(2002, 2003, 2004)
    For I = LBound(Months) To UBound(Months)
        Sheet1.ComboBox1.AddItem Months(I)
    Next I

    Sheet1.ComboBox2.List = WorksheetFunction.Transpose(Years)

End Sub
```

Метод AddItem объекта ComboBox добавляет значения из массива Months в список элемента управления ComboBox1. Для демонстрации альтернативной методики функция листа Transpose преобразовывает массив Years из строки в столбец, а значения этого столбца присваиваются свойству List объекта ComboBox2.

Обратите внимание, что для описания положения раскрывающихся списков используется имя листа Sheet1 (это имя можно увидеть в окне Project Explorer (Окно проекта) или в окне Свойства (Properties) в редакторе VBE). Несмотря на то что лист называется Database, он имеет программное имя Sheet1. Для изменения программного имени необходимо изменить значение (Name) (а не только значение Name) в окне Свойства (Properties) (рис. 23.6).

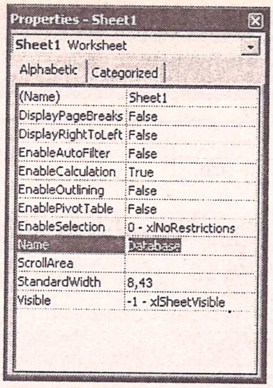


Рис. 23.6. Свойства листа. Видимое и программное имя листа

Следующий код добавляется в модуль кода листа:

```
Private Sub ComboBox1_Change()
    Call FilterDates
End Sub

Private Sub ComboBox2_Change()
    Call FilterDates
End Sub
```

При щелчке на раскрывающемся списке запускается подпрограмма `FilterDates`. Она может находиться в том же модуле и быть объявлена как `Private`, если ее не должны использовать другие модули. Кроме этого, подпрограмму `FilterDates` можно хранить в стандартном модуле и использовать в качестве универсальной вспомогательной процедуры.

Как написать подпрограмму `FilterDates`? Как было показано в предыдущих главах, можно применить механизм записи макросов. Это даст начальный фрагмент кода, с которым можно будет работать. После этого код будет модифицироваться для повышения производительности и гибкости. Следующий код получается при использовании механизма записи во время фильтрации дат:

```
Sub Macro4()
'
' Macro4 Macro
' Макрос записан 11/16/2003 Полом Киммелом
'

    Range("A5").Select
    Selection.AutoFilter
    Selection.AutoFilter Field:=1, Criteria1:=">=8/1/2003",
↳Operator:=xlAnd, _
    Criteria2:="<=9/15/2003"
End Sub
```

Как обычно, механизм записи макросов выделяет диапазоны и генерирует предельно точный код. Интерес представляет немного более универсальный код, который работает с именованным диапазоном, а не с конкретным диапазоном ячеек. В процессе преобразования кода можно использовать именованный диапазон `Database`.

```
Public Sub FilterDates()
    Range("Database").AutoFilter Field:=1, _
        Criteria1:=">=8/1/2003", _
        Operator:=xlAnd, Criteria2:="<9/15/2003"
End Sub
```

На основе записанного макроса можно создать универсальное решение. Например, необходимо выбрать месяц из первого раскрывающегося списка и год из второго раскрывающегося списка. После этого в диапазон добавляется один месяц, что дает начальную дату в виде выбранного года и месяца и конечную в виде года и месяца, следующего за выбранным. Результат преобразования кода показан ниже:

```
Private Sub FilterDates()
    Dim StartMonth As Integer
    Dim StartYear As Integer
    Dim StartDate As Date
    Dim EndDate As Date
    Dim StartCriterion As String

    Dim EndCriterion As String

    StartMonth = Me.ComboBox1.ListIndex + 1
    StartYear = Me.ComboBox2.Value
    StartDate = DateSerial(StartYear, StartMonth, 1)
    EndDate = DateSerial(StartYear, StartMonth + 1, 1)

    StartCriterion = ">=" & Format(StartDate, "mm/dd/yyyy")
    EndCriterion = "<" & Format(EndDate, "mm/dd/yyyy")

  Range("Database").AutoFilter Field:=1, Criteria1:=StartCriterion, _
    Operator:=xlAnd, Criteria2:=EndCriterion
End Sub
```

Подпрограмма `FilterDates` присваивает значения из раскрывающегося списка переменным `StartMonth` и `StartYear`. Вместо использования имени объекта `Sheet1` применяется ключевое слово `Me`, которое возвращает ссылку на лист, содержащий код. Это ключевое слово позволяет копировать код на другой лист не меняя имени листа.

Для определения значения переменной `StartMonth` применяется свойство `ListIndex` объекта `ComboBox1` (это свойство возвращает номер месяца). Так как свойство `ListIndex` начинает отсчет элементов с нуля, для получения правильного номера месяца к значению свойства добавляется 1. Функция `DateString` выполняет преобразование номеров года и месяца в строку даты, которая присваивается переменной `StartDate`. Во второй функции `DateSerial` подсчитывается дата на один месяц больше даты в переменной `StartDate`. Полученная дата присваивается переменной `EndDate`.

Функция `Format` выполняет преобразование значений переменных `StartDate` и `EndDate` обратно в строки, соответствующие региональным стандартам США (мм/дд/гггг). К строкам добавляются соответствующие логические операторы, и полученные строки присваиваются переменным `StartCriterion` и `EndCriterion` соответственно. Наконец, вызывается метод `AutoFilter` диапазона `Database`, при вызове которого используется сформированный критерий.

Получение точной даты

В работе функции `AutoFilter` есть еще одна проблема, возникающая при работе с датами во всех языковых версиях Excel. Она появляется при попытке получения точной даты, а не дат в определенном диапазоне. В этом случае функция `AutoFilter` пыта-

ется сравнивать критерий со строковым представлением даты на листе, а не с соответствующим значением даты.

Excel хранит даты в виде числовых значений. Дата хранится в виде количества дней с 1 января 1900 года. Например, 1 января 2003 года хранится в виде числа 37622. При запросе даты больше или равной 1 января 2003 года, Excel пытается найти даты больше или равные 37622. Но при запросе даты, равной 1 января 2003 года, Excel не ищет числовое представление даты. Вместо этого Excel пытается найти строковое представление даты 1 января 2003 на листе.

Следующая модификация подпрограммы `FilterDates` позволяет извлекать точную дату из списка, так как переменной `ExactCriterion` присваивается значение даты в формате "ммм дд, гггг". На листе даты отформатированы в виде "ммм дд, гггг":

```
Private Sub FilterExactDate()
    Dim ExactMonth As Integer
    Dim ExactYear As Integer
    Dim ExactDate As Date
    Dim ExactCriterion As String

    ExactMonth = Sheet1.ComboBox1.ListIndex + 1

    ExactYear = Sheet1.ComboBox2.Value
    ExactDate = DateSerial(ExactYear, ExactMonth, 1)
    ExactCriterion = Format(ExactDate, "mmm dd,yyyy")
    Range("Database").AutoFilter Field:=1, Criteria1:=ExactCriterion
End Sub
```

Обратите внимание, что при изменении формата даты, который используется на листе, придется изменить формат даты и в коде, иначе код не будет работать.

Предыдущий код будет выдавать все записи для первого месяца, так как значение 1 передается в качестве третьего параметра функции `DateSerial`. Для выбора любой даты в пределах месяца в ячейку A2 необходимо добавить третий раскрывающийся список. Кроме этого, в процедуру обработки события `ComboBox1_Click` нужно ввести код, который будет добавлять в список соответствующее количество дней в зависимости от месяца, выбранного в раскрывающемся списке `ComboBox1`.

Копирование видимых строк

Для более простого создания листа с копией отфильтрованных данных в начале листа достаточно добавить кнопку ActiveX. В модуль листа можно ввести подпрограмму `Click`. Эта процедура копирует весь диапазон `Database`, так как по умолчанию копируются только видимые строки.

```
Private Sub CommandButton1_Click()
    Dim NewWorksheet As Worksheet
    Dim WorksheetName As String
    Dim Month As String
    Dim Year As String
    Dim TempWorksheet As Worksheet

    Month = Sheet1.ComboBox1.Value
    Year = Sheet1.ComboBox2.Value
    On Error Resume Next
    WorksheetName = Format(_
      DateValue(Year & " " & Month & " 1"), "mmm yyyy")
```

```
    Set TempWorksheet = Worksheets(WorksheetName)

    If Err.Number = 0 Then
        MsgBox "Эти данные уже копировались"
        Exit Sub
    End If

    On Error GoTo 0

    Set NewWorksheet = Worksheets.Add
    Range("Database").Copy Destination:=NewWorksheet.Range("A1")
    NewWorksheet.Columns("A:G").AutoFit
    NewWorksheet.Name = WorksheetName
End Sub
```

Процедура обработки события Click вычисляет имя нового листа в формате "ммм гггг". После этого проверяется существование такого листа. Объектной переменной присваивается ссылка на лист с новым именем. Если при этом не возникает ошибка, лист уже существует. В таком случае процедура выдает сообщение и завершает работу.

Если листа с новым именем нет, процедура обработки события добавляет новый лист после существующих листов. Диапазон Database копируется на новый лист. Ширина столбцов автоматически меняется в соответствии с размером скопированных данных. После этого процедура присваивает имя новому листу.

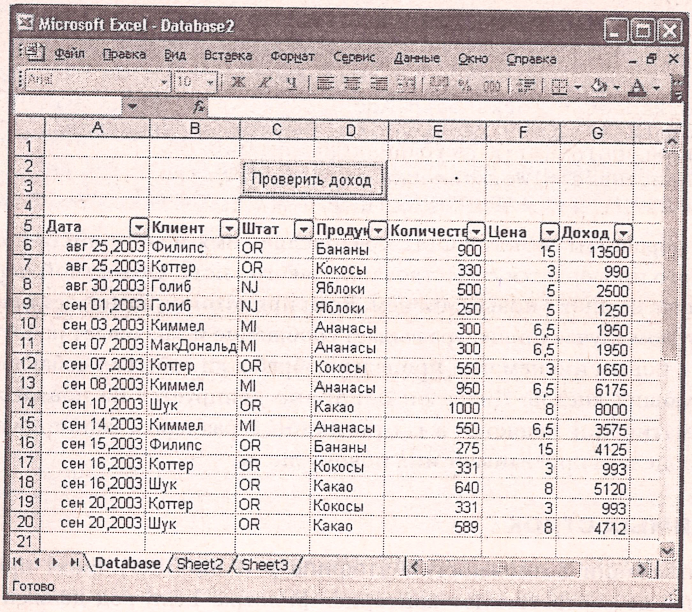
Поиск видимых строк

При использовании возможности Автофильтр (AutoFilter) Excel просто скрывает строки, которые не соответствуют критерию фильтра. Если код должен обрабатывать только видимые строки, необходимо просмотреть все строки в списке и определить, скрыта ли каждая конкретная строка. С этой операцией связана определенная сложность. При использовании свойства Hidden объекта Range этот объект должен представлять целую строку (от столбца A до IV) или целый столбец от строки 1 до 65536. Свойство Hidden единственной ячейки или строки из 7 столбцов (рис. 23.7) применять нельзя.

Следующий код проверяет видимость каждой строки и закрашивает фон тех строк, в которых неправильно подсчитывается значение дохода.

```
Private Sub CommandButton1_Click()
    Dim Database As Range
    Dim Row As Range

    Dim NumberSold As Double
    Dim Price As Double
    Dim Revenue As Double

    With Range("Database")
        Set Database = .Offset(1, 0).Resize(.Rows.Count - 1)
    End With

    For Each Row In Database.Rows
        If Row.EntireRow.Hidden = False Then
            NumberSold = Row.Cells(5).Value
            Price = Row.Cells(6).Value
            Revenue = Row.Cells(7).Value
                If Abs(NumberSold * Price - Revenue) > 0.000001 Then
                    Row.Select
                    MsgBox "Ошибка в выделенной строке"
                    Row.Interior.ColorIndex = 15
```

```
                End If
        End If
    Next Row
End Sub
```



	A	B	C	D	E	F	G
5	Дата	Клиент	Штат	Продук	Количеств	Цена	Доход
6	авг 25,2003	Филипс	OR	Бананы	900	15	13500
7	авг 25,2003	Коттер	OR	Кокосы	330	3	990
8	авг 30,2003	Голиб	NJ	Яблоки	500	5	2500
9	сен 01,2003	Голиб	NJ	Яблоки	250	5	1250
10	сен 03,2003	Киммел	MI	Ананасы	300	6,5	1950
11	сен 07,2003	МакДональд	MI	Ананасы	300	6,5	1950
12	сен 07,2003	Коттер	OR	Кокосы	550	3	1650
13	сен 08,2003	Киммел	MI	Ананасы	950	6,5	6175
14	сен 10,2003	Шук	OR	Какао	1000	8	8000
15	сен 14,2003	Киммел	MI	Ананасы	550	6,5	3575
16	сен 15,2003	Филипс	OR	Бананы	275	15	4125
17	сен 16,2003	Коттер	OR	Кокосы	331	3	993
18	сен 16,2003	Шук	OR	Какао	640	8	5120
19	сен 20,2003	Коттер	OR	Кокосы	331	3	993
20	сен 20,2003	Шук	OR	Какао	589	8	4712

Рис. 23.7. Поиск видимых строк

В процедуре обработки события `Click` командной кнопки сначала определяется объектная переменная `Database`, которая ссылается на диапазон `Database` (кроме строки заголовков). После этого все строки диапазона `Database` обрабатываются в цикле `For Each... Next`. Первый оператор `If` обеспечивает обработку только видимых строк. Значения полей Количество, Цена и Доход из текущей строки присваиваются переменным, и второй оператор `If` проверяет допустимость значения поля Доход (равное произведению полей Количество и Цена).

> *Так как вычисления на листе выполняются с использованием двоичного представления с сохранением точности до 15 значащих цифр, равенство двух значений не всегда стоит проверять до последней десятичной точки, особенно, если значения извлекаются из другого листа. Вместо этого стоит проверять, выходит ли разница между значениями за допустимые пределы. Так как разность значений может быть отрицательной и положительной, для преобразования разности в положительную используется функция* `Abs`.

Если проверка показывает недопустимую разницу между значениями, строка выделяется и выдается соответствующее сообщение. Кроме этого, в строке устанавливается светло-серый цвет фона.

Расширенный фильтр

Самым мощным способом фильтрации данных в списке является использование возможности Расширенный фильтр (Advanced Filter). Список можно фильтровать на месте с помощью автофильтра, а можно копировать в другое место. Этим местом может быть тот же лист, другой лист в пределах той же книги или другая открытая книга. В следующем примере извлекаются данные для штатов OR и MI за третий квартал 2003 года. Данные копируются в новую книгу.

При использовании расширенного фильтра критерии указываются в диапазоне листа. Например, на рис. 23.8 показан диапазон `Criteria` (A1:C3). Эта книга называется `Extract.xls`. Список данных находится в книге `Database1.xls`. В ней хранятся те же данные, которые использовались в примерах автофильтров.

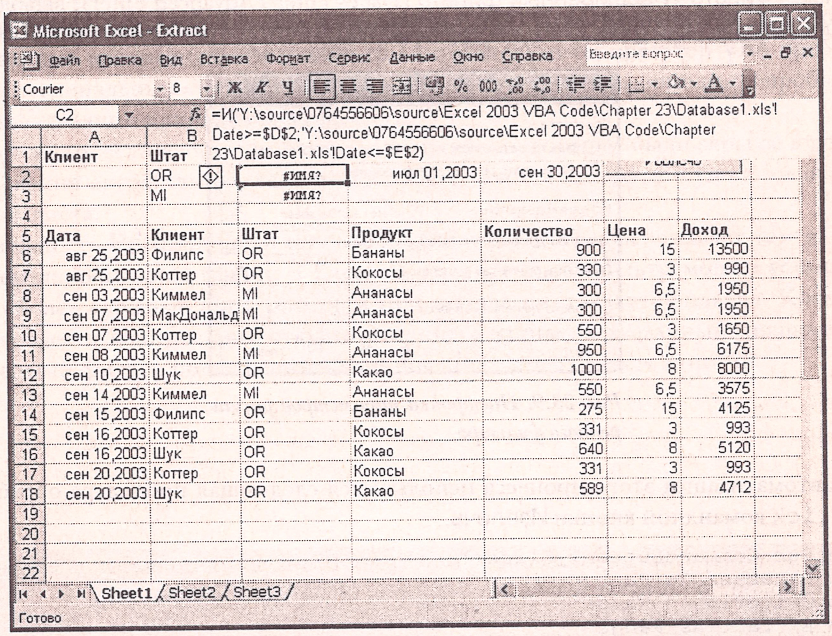


Рис. 23.8. Диапазон с критериями

В верхней строке диапазона `Criteria` указываются названия полей из фильтруемого списка. Под названиями полей может быть расположено любое количество строк. Критерии в разных строках комбинируются с помощью оператора OR. Критерии в пределах одной строки комбинируются с помощью оператора AND. Кроме этого, можно использовать вычислимые критерии в форме логических операторов, которые преобразовываются в значения `True` или `False`. При применении вычислимых критериев верхняя строка диапазона `Criteria` должна быть пустой или содержать метку, которая не является названием поля в списке (в данном случае это Вычисление).

При создании вычислимого критерия в формулах можно использовать названия полей из списка данных. В строке формулы показано содержимое ячейки C2:

```
=AND(Database1.xls!Date>=$D$2,Database1.xls!Date<=$E$2)
```

Формула в ячейке C3 ничем не отличается от формулы в ячейке C2.

Показанные критерии эквиваленты применению следующего фильтра:

```
(State=OR AND Date>=Jul 1,2003 AND Date<Sep 30,2003) OR _
   (State=MI AND Date>=Jul 1,2003 AND Date<Sep 30,2003)
```

Так как список с данными находится в книге `Database1.xls`, ссылки на поле Дата вводятся как внешние. В данном случае используются такие же ссылки, как и при обращении к именам в другой книге. Так как названия полей не являются именами книги, формула возвращает значение `#NAME?`.

Для использования возможности Расширенный фильтр (Advanced Filter) список данных в книге `Database1.xls` должен быть переименован в `Database`. В книге `Extract.xls` диапазон A1:C3 переименован в `Criteria`, а диапазон A5:G5 преобразован в `Extract`. Если возможность Расширенный фильтр (Advanced Filter) запускается вручную с помощью команды Данные⇨Фильтр⇨Расширенный фильтр (Data⇨Filter⇨Advanced Filter), выводится следующее диалоговое окно (рис. 23.9), в котором можно ввести соответствующие имена.

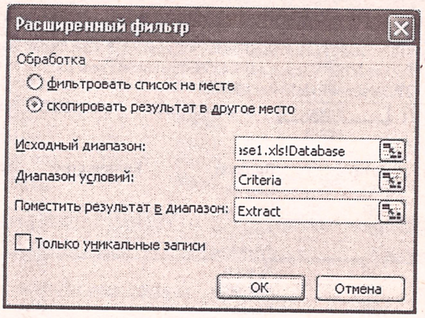


Рис. 23.9. Настройка параметров расширенного фильтра

Для автоматизации этого процесса используется следующая процедура обработки события `Click` командной кнопки Извлечь.

```
Private Sub CommandButton1_Click()
    Dim Database As Range
    Dim Criteria As Range
    Dim Extract As Range

    Set Database = Workbooks("Database1.xls"). _
        Worksheets("Database").Range("Database")

    Set Criteria = ThisWorkbook.Worksheets(1).Range("Criteria")
    Set Extract = ThisWorkbook.Worksheets(1).Range("Extract")
    Database.AdvancedFilter Action:=xlFilterCopy, _
        CriteriaRange:=Criteria, CopyToRange:=Extract

End Sub
```

В процедуре обработки события определяются три объектные переменные, которые ссылаются на диапазоны `Database`, `Criteria` и `Extract`. После этого вызывается метод `AdvancedFilter` объекта `Range`, который называется `Database`.

Резюме

Возможность Данные⇨Форма (Data⇨Form) значительно упрощает создание макросов для управления данными. Но если верхний левый угол списка находится за пределами диапазона A1:B2, список с данными необходимо переименовать в`Database`.

Как было показано выше, возможности Автофильтр (AutoFilter) и Расширенный фильтр (Advanced Filter) могут использоваться вместе с кодом VBA для получения гибких решений по извлечению данных из списков. Комбинирование этих возможностей и элементов управления ActiveX (раскрывающихся списков и кнопок) позволяет сделать их доступными для пользователей всех уровней подготовки. Для получения списка необходимых методов можно воспользоваться механизмом записи макросов. После этого записанный код адаптируется для применения элементов управления ActiveX.

При этом необходимо обратить особое внимание на использование форматов даты, не соответствующих региональным стандартам США. При сравнении диапазонов с датами с помощью возможности Автофильтр (AutoFilter) интерпретатор VBA требует применения региональных стандартов США. Дополнительная информация о проблемах интернационализации приводится в главе 17.

Кроме этого, если необходимо определить строки, которые были скрыты в результате использования автофильтра, свойство `Hidden` объекта `Range` доступно только для целых строк листа.

Возможность Расширенный фильтр (Advanced Filter) обеспечивает самую мощную функциональность фильтрации в Excel. В расширенный фильтр можно устанавливать намного более сложные критерии, чем при использовании автофильтра. Кроме этого, поддерживается копирование отфильтрованных данных в указанный диапазон. Также расширенный фильтр поддерживает копирование данных между книгами.

24

Генерация диаграмм

В этой главе рассматривается использование механизма записи макросов для определения объектов, методов и свойств, позволяющих управлять диаграммами. В дальнейшем записанный код улучшается и расширяется для получения большей гибкости и эффективности. Основное внимание уделяется доступу к объекту Chart из кода VBA. Этот объект предоставляет программный доступ к большому количеству объектов, содержащихся в диаграммах Excel. Дополнительная информация об этих объектах приводится в приложении А. В частности, в этой главе будут рассматриваться следующие темы:

- создание объектов Chart на отдельных листах;
- создание встроенных в лист объектов Chart;
- редактирование рядов на диаграммах;
- определение рядов с помощью массивов;
- определение меток диаграмм, связанных с ячейками на листах.

В Excel поддерживается создание диаграмм двух типов: диаграмм на отдельных листах и диаграмм, встроенных в существующие листы. Оба типа управляются одинаково. Единственное отличие заключается в том, что лист диаграммы является отдельным объектом Chart, а встроенная диаграмма хранится в объекте ChartObject. Каждый объект ChartObject на листе электронной таблицы является членом коллекции ChartObjects. Листы диаграмм входят в коллекцию Charts объекта Workbook.

Каждый объект ChartObject является членом коллекций Shapes и ChartObjects. Коллекция Shapes предоставляет альтернативный способ получения ссылок на встроенные диаграммы. Механизм записи макросов генерирует код, использующий коллекцию Shapes.

Листы диаграмм

Перед созданием диаграммы включите механизм записи макросов. С помощью мастера диаграмм создайте новый лист диаграммы. Диаграмма должна отображать данные из диапазона A3:D7 (рис. 24.1). На втором шаге работы мастера выберите переключатель Ряды в столбцах (Series in Rows). На третьем шаге назовите диаграмму Манго. На четвертом шаге выберите переключатель Отдельном (As new sheet) и введите имя листа Манго.

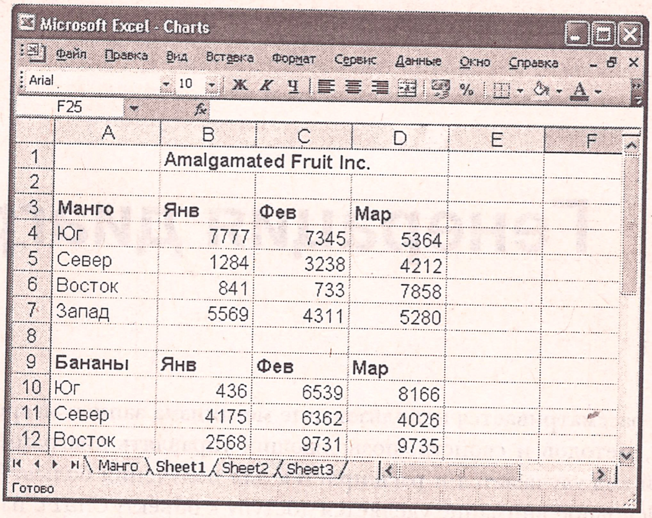


Рис. 24.1. Диапазон с данными для диаграммы

Записанный макрос

Записанный макрос должен выглядеть следующим образом:

```
Sub Макрос1()
'
' Макрос1 Макрос
' Макрос записан 31.03.2005 (Leschinsky Oleg)
'

    Charts.Add
    ActiveChart.ChartType = xlColumnClustered
    ActiveChart.SetSourceData Source:=Sheets("Sheet1").Range("A3:D7"), _
PlotBy:= xlRows
    ActiveChart.Location Where:=xlLocationAsNewSheet, Name:="Манго"
    With ActiveChart
        .HasTitle = True
        .ChartTitle.Characters.Text = "Манго"
        .Axes(xlCategory, xlPrimary).HasTitle = False
        .Axes(xlValue, xlPrimary).HasTitle = False
    End With
End Sub
```

В записанном макросе для создания диаграммы используется метод `Add` коллекции `Charts`. Сначала определяется свойство `ChartType` активной диаграммы, после чего метод `SetSourceData` определяет диапазон с данными для диаграммы. Метод `Location` используется для определения листа диаграммы и присвоения ему имени. Свойство `HasTitle` устанавливается в значение `True`, чтобы можно было определить значение свойства `ChartTitle`. Наконец, свойство `HasTitle` осей устанавливается в значение `False` (этот шаг является необязательным).

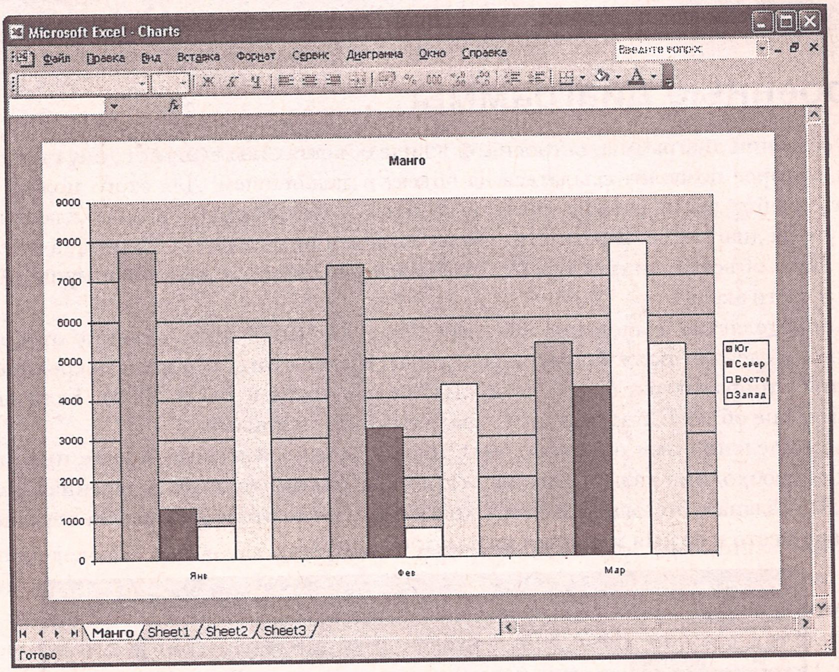


Рис. 24.2. Диаграмма, созданная в результате работы макроса

Добавление листа диаграммы с помощью кода VBA

Записанный код неплохо выглядит даже без модификации. Но лучше создавать объектную переменную, которая обеспечивает возможность ссылаться на диаграмму в последующем коде. Кроме этого, можно удалить избыточный код и добавить заголовок диаграммы, связанный с листом. Эти изменения отражены в следующем коде:

```
Option Explicit

Public Sub AddChartSheet()
    Dim aChart As Chart

    Set aChart = Charts.Add    With aChart
        .Name = "Манго"
        .ChartType = xlColumnClustered
        .SetSourceData Source:=Sheets("Sheet1").Range("A3:D7"),
↳PlotBy:=xlRows
        .HasTitle = True
        .ChartTitle.Text = "=Sheet1!R3C1"
  End With
End Sub
```

Вызов метода `Location` был удален, так как в данном случае он не нужен, потому что лист диаграммы создается по умолчанию. Мастер диаграмм не позволяет ввести формулу, определяющую заголовок диаграммы. Но изменение заголовка диаграммы можно записать отдельно. Из записанного кода станет понятно, что для этого свойству `Text` объекта `ChartTitle` необходимо присвоить текст формулы. В показанном выше коде заголовок диаграммы определяется как формула, ссылающаяся на значение в ячейке R3C1 или A3.

> При вводе формулы в текстовый элемент диаграммы необходимо использовать адресацию по номерам строк и столбцов (R3C1), а не по именам ячеек (A3).

Встроенные диаграммы

При создании диаграммы, встроенной в виде объекта `ChartObject`, ему стоит присвоить имя, которое позволит ссылаться на объект в дальнейшем. Для этого можно вручную выделить ячейку листа (чтобы снять выделение с диаграммы), нажать клавишу <Ctrl> и щелкнуть на диаграмме. В результате будет выделен объект `ChartObject`, а не сама диаграмма. Имя объекта `ChartObject` отображается слева от поля **Формула** (Formula) в верхней части экрана.

Как определяется выделение объекта `ChartObject`? Имя объекта отображается в поле имени слева от поля **Формула** (Formula). Кроме этого, в углах и посредине граней диаграммы отображаются белые квадраты (белые круги в Excel 2003). Если выделить диаграмму, а не объект `ChartObject`, квадраты будут черными.

После выделения имя объекта `ChartObject` можно изменить в поле имени. После изменения необходимо нажать клавишу <Enter>. Показанная ниже встроенная диаграмма (рис. 24.3) создана с помощью мастера диаграмм. После этого диаграмма перетаскивается на новое место и ее имя меняется на `MangoesChart`.

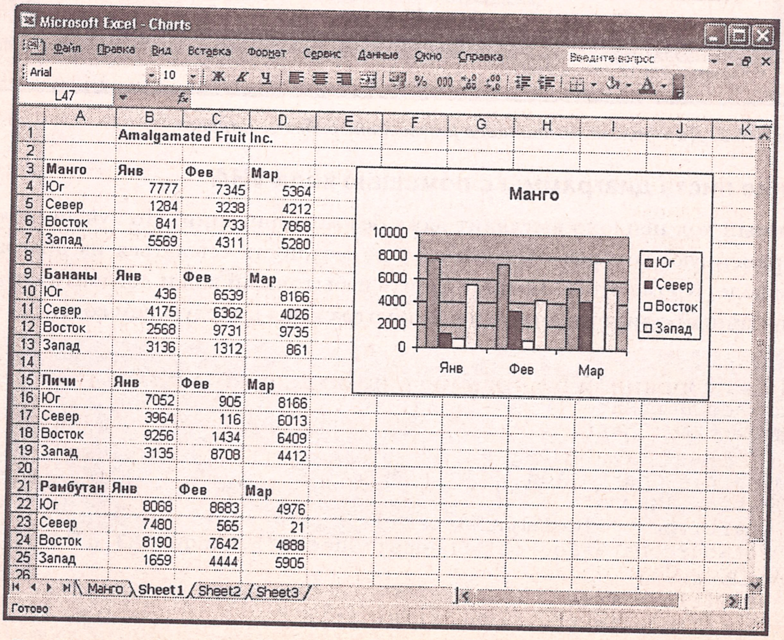


Рис. 24.3. Создание встроенной диаграммы из кода

Использование механизма записи макросов

При выделении диапазона A3:D7 и включении механизма записи макросов до создания диаграммы, включая перемещение объекта ChartObject на нужное место и изменение имени на MangoesChart, будет записан следующий код:

```
Sub Макрос2()
'
' Макрос2 Макрос
' Макрос записан 31.03.2005 (Leschinsky Oleg)
'
'
    Charts.Add
    ActiveChart.ChartType = xlColumnClustered
    ActiveChart.SetSourceData
↳ Source:=Sheets("Sheet1").Range("A3:D7"), PlotBy:= xlRows
    ActiveChart.Location Where:=xlLocationAsObject, Name:="Sheet1"
    With ActiveChart
        .HasTitle = True
        .ChartTitle.Characters.Text = "Манго"
        .Axes(xlCategory, xlPrimary).HasTitle = False
        .Axes(xlValue, xlPrimary).HasTitle = False
    End With
    ActiveSheet.Shapes("Диагр. 27").IncrementLeft 96#
    ActiveSheet.Shapes("Диагр. 27").IncrementTop -27.75
    ActiveWindow.Visible = False
    Windows("Charts.xls").Activate
    Range("G4").Select
    ActiveSheet.Shapes("Chart 27").Select
    Selection.Name = "MangoesChart"
End Sub
```

До определения заголовка диаграммы записанный макрос напоминает макрос для создания листа диаграммы. Создание встроенной диаграммы обеспечивается методом Location. До оператора EndWith записанный код можно использовать повторно. Но код перемещения объекта ChartObject и изменение имени объекта для повторного применения не подходят. В этом коде для идентификации объекта ChartObject используется имя, принятое по умолчанию. (Обратите внимание, что механизм записи макросов ссылается на объект ChartObject как на объект Shape. Это отличие было описано в начале главы.)

При попытке повторного запуска этого кода или при адаптации кода для другой диаграммы код откажет при обращении к объекту Chart30 или к другой диаграмме, которая создавалась при записи макроса. Способ обращения к объекту ChartObject и получения ссылки на активный лист диаграммы является совершенно неочевидным.

Добавление встроенной диаграммы с помощью кода VBA

В следующем коде свойство Parent встроенной диаграммы используется для идентификации объекта ChartObject, внутри которого хранится диаграмма:

```
Sub AddChart()
    Dim aChart As Chart

    ActiveSheet.ChartObjects.Delete
        Set aChart = Charts.Add
        Set aChart = aChart.Location(Where:=xlLocationAsObject,
↳Name:="Sheet1")
    With aChart
```

```
        .ChartType = xlColumnClustered
        .SetSourceData Source:=Sheets("Sheet1").Range("A3:D7"), _
            PlotBy:=xlRows
        .HasTitle = True
        .ChartTitle.Text = "=Sheet1!R3C1"
        With .Parent
            .Top = Range("F3").Top
            .Left = Range("F3").Left
            .Name = "MangoesChart"
        End With
    End With
End Sub
```

Сначала подпрограмма AddChart удаляет существующие объекты ChartObjects. После этого объектной переменной aChart присваивается ссылка на добавленную диаграмму. По умолчанию новая диаграмма находится на листе диаграммы, поэтому метод Location применяется для превращения диаграммы во встроенную диаграмму.

> При использовании метода Location объекта Chart объект Chart удаляется и создается заново. При этом удаляются все ссылки на оригинальный объект Chart. Таким образом, для получения ссылки на новый объект Chart возвращаемое значение метода Location необходимо присваивать объектной переменной типа Chart.

Подпрограмма AddChart определяет тип диаграммы, источник данных и заголовок диаграммы, представляющий собой формулу, ссылающуюся на ячейку A3. Подпрограмма AddChart устанавливает свойства Top и Left объекта ChartObject равными свойствам Top и Left ячейки F3. Для получения ссылки на объект ChartObject подпрограмма AddChart использует свойство Parent объекта Chart. Наконец, подпрограмма AddChart присваивает объекту ChartObject новое имя, что позволит ссылаться на объект в будущем.

Редактирование рядов данных

Самым быстрым способом определения полностью нового набора данных для диаграммы является использование метода SetSourceData объекта Chart. Кроме этого, отдельными рядами можно управлять с помощью объекта Series, который хранится в объекте SeriesCollection в пределах диаграммы. В следующем примере показано, как получать доступ к отдельным рядам.

В данном случае из диаграммы MangoesChart удаляются все ряды, после чего по одному добавляются четыре новых. Новая диаграмма будет содержать информацию о продуктах в выбранном пользователем регионе. Для упрощения поиска данных о продуктах каждому диапазону продукта присвоены имена. Например, диапазону A3:D7 присвоено имя Mangoes (соответствующее метке в ячейке A3). Полученная в результате диаграмма показана на рис. 24.4.

Следующий код выполняет преобразование диаграммы MangoesChart и добавляет новые данные (обратите внимание, что для работы кода на листе должна присутствовать оригинальная диаграмма). Так как подпрограмма MangoesToRegion состоит из нескольких частей, рассмотрим каждую часть в отдельности. (Здесь приводится полный листинг подпрограммы MangoesToRegion. Вспомогательные методы будут рассматриваться далее.)

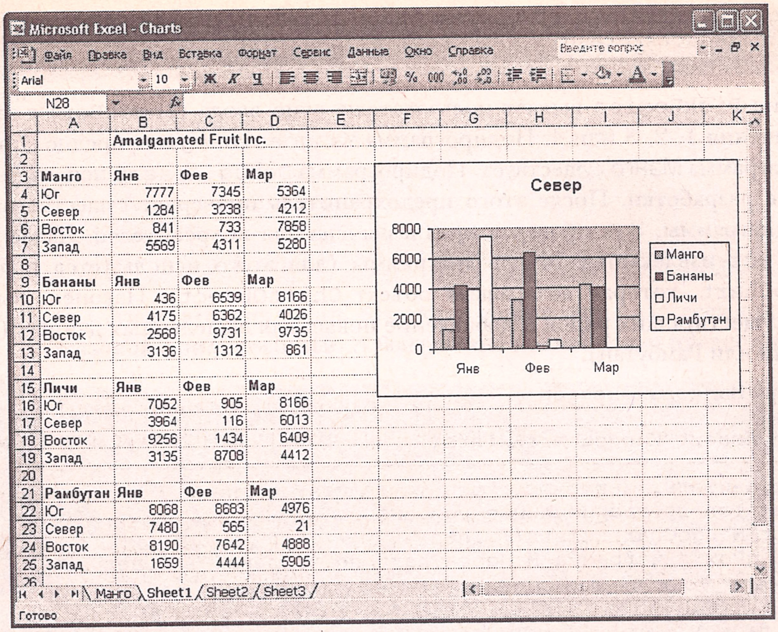


Рис. 24.4. Добавление рядов на диаграмму

```
Public Function PickRegion() As Integer

  Dim Answer As String

  Do
    Answer = InputBox("Введите номер Региона (1 to 4)")
    If Answer >= "1" And Answer <= "4" Then
      Exit Do
    Else
      Call MsgBox("Выберите номер Региона из 1, 2, 3 или 4", vbCritical)
    End If
  Loop

  PickRegion = CInt(Answer)
End Function

Public Sub MangoesToRegion()

  Debug.Assert MangoChartExists()

  ' Предохранитель: Нет смысла что-то делать, если не
  ' выполняется предварительное условие
    If Not MangoChartExists() Then
    MsgBox "Диаграмма MangoesChart не найдена - процедура завершена",
↳ vbCritical
    Exit Sub
  End If

  Dim aChart As Chart
  Set aChart = GetMangoChart().Chart
  Debug.Assert (aChart Is Nothing = False)
```

```
    Call DeleteSeries(aChart.SeriesCollection)
    Call AddProductsForRegion(PickRegion(), aChart)
End Sub
```

Функция PickRegion выдает пользователю окно ввода (InputBox) с запросом на выбор значения 1, 2, 3 или 4. Подпрограмма MangoesToRegion предполагает, что исходная диаграмма Манго существует. Подпрограмма Debug.Assert используется только в процессе разработки. После этого предохранитель проверяет существование оригинальной диаграммы. Эта проверка предназначена для потребителей. Далее из объекта ChartObject извлекается объект диаграммы (для этого используется подпрограмма GetMangoChart, которая возвращает объект ChartObject). Наконец, выполняется удаление существующих рядов и добавление новых для выделенных данных (Манго, Бананы, Личи или Рамбутан):

```
Public Sub MangoesToRegion()

    Debug.Assert MangoChartExists()

 ' Предохранитель: Нет смысла что-то делать, если не
 ' выполняется предварительное условие
    If Not MangoChartExists() Then
        MsgBox "Диаграмма MangoesChart не найдена - процедура
⇘завершена", vbCritical
        Exit Sub
    End If

    Dim aChart As Chart
    Set aChart = GetMangoChart().Chart
    Debug.Assert (aChart Is Nothing = False)

    Call DeleteSeries(aChart.SeriesCollection)
    Call AddProductsForRegion(PickRegion(), aChart)
End Sub
```

Работа этого кода обеспечивается методами MangoChartExists, GetMangoChart, DeleteSeries, AddProductsForRegion и PickRegion. Рассмотрим каждый из этих методов более подробно:

```
Public Function MangoChartExists() As Boolean
    MangoChartExists = GetMangoChart Is Nothing = False
End Function
```

Подпрограмма MangoChartExists вызывает подпрограмму GetMangoChart и проверяет действительность возвращаемого объекта ChartObject. Если диаграмма не существует, обработчик ошибок в подпрограмме GetMangoChart возвращает значение Nothing.

```
Public Function GetMangoChart() As ChartObject
    On Error GoTo Catch
    Set GetMangoChart = Worksheets("Sheet1").
⇘ChartObjects("MangoesChart")
    Exit Function

Catch:
    Set GetMangoChart = Nothing
End Function
```

После этого подпрограмма DeleteSeries принимает параметр типа SeriesCollection, который извлекается из объекта Chart. Далее подпрограмма перебирает и удаляет существующие ряды.

```
Public Sub DeleteSeries(ByVal Series As SeriesCollection)

    Dim I As Integer
    For I = Series.Count To 1 Step -1
        Series(I).Delete
    Next I
End Sub
```

После удаления существующих рядов создаются новые ряды для всех продуктов из выделенных диапазонов с информацией. В завершение новым рядам предоставляется заголовок и имя:

```
Public Sub AddProductsForRegion(ByVal region As Integer, _
  ByVal Chart As Chart)

    Dim I As Integer
    Dim J As Integer
    Dim rangeY As Range
    Dim rangeX As Range
    Dim products As Variant
    Dim regions As Variant
    Dim productsname As Variant
    Dim regionsname As Variant

    products = Array("Mangoes", "Bananas", "Lychees", "Rambutan")
    regions = Array("South", "North", "East", "West")


    productsname = Array("Манго", "Бананы", "Личи", "Рамбутан")
    regionsname = Array("Юг", "Север", "Восток", "Запад")

    For I = LBound(products) To UBound(products)
      Set rangeY = Range(products(I)).Offset(region, 1).Resize(1, 3)
      Set rangeX = Range(products(I)).Offset(0, 1).Resize(1, 3)
      With Chart.SeriesCollection.NewSeries
          .Name = productsname(I)
          .Values = rangeY
          .XValues = "=" & rangeX.Address _
              (RowAbsolute:=True, _
              ColumnAbsolute:=True, _
              ReferenceStyle:=xlR1C1, _
              External:=True)
      End With
    Next I

    Chart.ChartTitle.Text = regionsname(region - 1 + LBound(regions))
    GetMangoChart().Name = "RegionChart"
End Sub
```

Вот и все. В двух словах это выглядит так: была найдена существующая диаграмма, на ней удалены существующие ряды и добавлены новые данные. Показанное в этом примере разбиение на методы является более предпочтительным, так как оно позволяет сократить объем ненужных комментариев и отказаться от использования большого количества временных локальных переменных.

Десятилетний опыт программирования позволяет писать большие монолитные методы с большим количеством временных переменных. Это один из допустимых подходов к программированию, но современные подходы к методам разработки программного обеспечения, включая *рефакторинг (refactoring)*, который может применяться в VBA как и в любом другом объектно-ориентированном языке, доказывают, что короткие методы, уменьшение количества временных переменных и правильно названные методы вместо большого количества временных переменных позволяют сделать код более простым для сопровождения. Дополнительная информация на эту тему приводится в книге Мартина Фаулера *Refactoring: Improving the Design of Existing Code* (Addison-Wesley, 1999).

Определение рядов диаграммы с помощью массивов

Для определения ряда диаграммы можно присвоить массив свойству `Values` объекта ряда. Такой способ определения ряда требуется для генерации диаграммы, не связанной с исходными данными. Подобная диаграмма может распространяться в отдельной книге независимо от исходных данных.

На рис. 24.5 показана диаграмма для данных из диапазона `Mangoes`. В поле **Формула** (Formula) над листом показано определение первого ряда данных. Названия месяцев и значения на вертикальной оси определяются в массивах Excel. Текст имен диапазонов присваивается именам рядов.

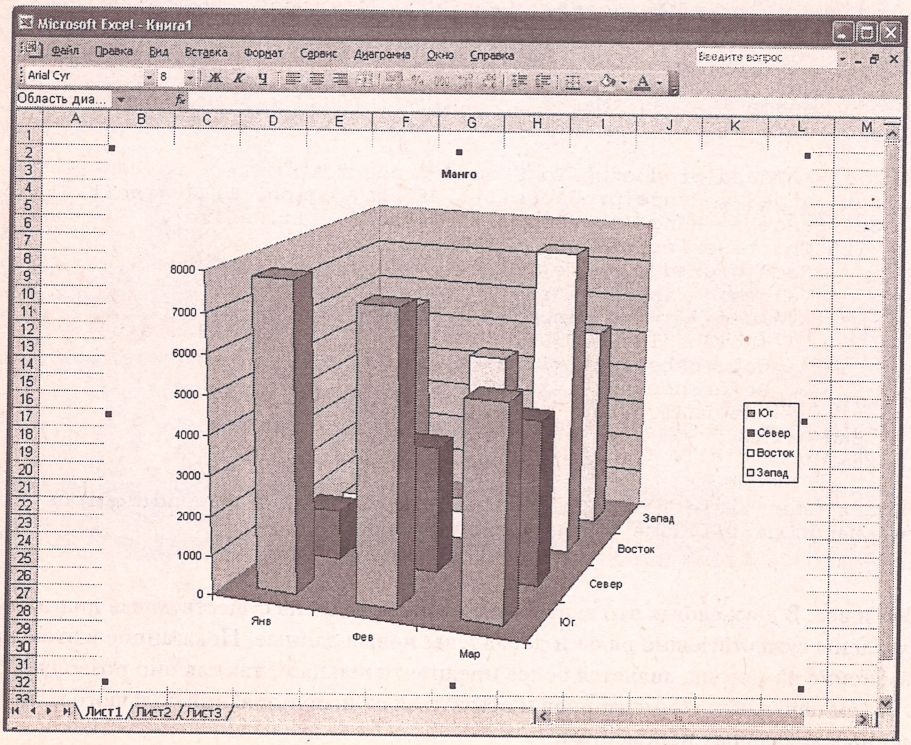


Рис. 24.5. Создание трехмерной диаграммы

> Функция SERIES поддерживает использование массивов до 250 элементов. Это ограничивает количество точек диаграммы, которые можно указать с помощью массива.

Следующий код позволяет создать трехмерную диаграмму:

```
Public Sub MakeArrayChart()
    Dim sourceWorksheet As Worksheet
    Dim sourceRange As Range
    Dim aWorkbook As Workbook
    Dim aWorksheet As Worksheet
    Dim aChart As Chart
    Dim aNewSeries As Series
    Dim I As Integer
    Dim SalesArray As Variant
    Dim MonthArray As Variant

    MonthArray = Array("Янв", "Фев", "Мар")

    ' Определить источник данных
    Set sourceWorksheet = ThisWorkbook.Worksheets("Sheet1")
    Set sourceRange = sourceWorksheet.Range("Mangoes")

    ' Создать новую книгу
    Set aWorkbook = Workbooks.Add
    Set aWorksheet = aWorkbook.Worksheets(1)

    ' Создать новый объект диаграммы и встроить его в лист
    Set aChart = aWorkbook.Charts.Add

    Set aChart = aChart.Location(Where:=xlLocationAsObject,
↳Name:="Лист1")

    With aChart
        ' Определить тип диаграммы
        .ChartType = xl3DColumn
        For I = 1 To 4
            ' Создать новый ряд
            Set aNewSeries = .SeriesCollection.NewSeries
            ' Передать данные в массиве
            SalesArray = sourceRange.Offset(I, 1).Resize(1, 3).Value
            aNewSeries.Values = SalesArray
            aNewSeries.XValues = MonthArray
            aNewSeries.Name = sourceRange.Cells(I + 1, 1).Value
        Next I

        .HasLegend = False
        .HasTitle = True
        .ChartTitle.Text = "Манго"
        ' Расположить и назвать объект ChartObject в диапазоне B2:I22
        With .Parent
            .Top = aWorksheet.Range("B2").Top
            .Left = aWorksheet.Range("B2").Left
            .Width = aWorksheet.Range("B2:I22").Width
            .Height = aWorksheet.Range("B2:I22").Height
            .Name = "ArrayChart"
        End With
    End With
End Sub
```

(Именованный диапазон Mangoes добавляется с помощью команды меню Вставка⇨Имя⇨Определить (Insert⇨Name⇨Define). Диапазон Mangoes определяется ячейками с A3 по D7.) Подпрограмма MakeArrayChart заносит названия месяцев в массив MonthArray. При необходимости эти данные могут извлекаться из листа. Ссылка на лист с данными присваивается переменной sourceWorksheet. Диапазон Mangoes сохраняется в переменной sourceRange. Для диаграммы создается новая книга и ссылка на эту книгу присваивается переменной aWorkbook. Ссылка на первый лист новой книги присваивается переменной aWorksheet. Новая диаграмма добавляется в коллекцию Charts в пределах объекта aWorkbook. Ссылка на новую диаграмму сохраняется в переменной aChart. Метод Location обеспечивает преобразование диаграммы во встроенную диаграмму и переопределяет значение переменной aChart.

В структуре With... End With с помощью свойства ChartType объекта aChart диаграмма превращается в трехмерную гистограмму. В цикле For... Next создается четыре новых последовательности. На каждой итерации цикла с помощью метода NewSeries создается новый ряд. Данные диапазона из соответствующей строки присваиваются непосредственно переменной SalesArray типа Variant. Сразу после этого значение переменной SalesArray присваивается свойству Values объекта нового ряда.

Массив MonthArray сохраняется в свойстве XValues объекта нового ряда. Текст в столбце A диапазона Mangoes присваивается свойству Name объекта нового ряда.

После этого код удаляет добавляемую по умолчанию легенду диаграммы и устанавливает заголовок диаграммы. Оставшийся код работает с объектом ChartObject, который является родительским объектом для объекта диаграммы. Этот объект размещается точно над ячейками B2:I22. Кроме этого, диаграмме назначается имя ArrayChart.

В результате работы кода создается диаграмма в новой книге. Полученная диаграмма полностью независима от исходной книги и содержащихся в ней данных. Если диаграмму скопировать и вставить в новую книгу, она все еще будет содержать исходные данные.

Преобразование диаграммы для использования массивов

В преобразовании существующей диаграммы для использования массивов, а не ссылок на ячейки, нет ничего сложного. После такого преобразования диаграмма становится независимой от данных, на основе которых она создавалась. В следующем фрагменте кода показан процесс такого преобразования:

```
Public Sub ConvertSeriesValuesToArrays()
    Dim aSeries As Series
    Dim aChart As Chart

    On Error GoTo Catch

    Set aChart = ActiveSheet.ChartObjects(1).Chart
    For Each aSeries In aChart.SeriesCollection
        aSeries.Values = aSeries.Values
        aSeries.XValues = aSeries.XValues
        aSeries.Name = aSeries.Name
    Next aSeries

    Exit Sub
Catch:
```

```
    MsgBox "Извините. Объем данных превышает ограничение на размер
↳массива"
End Sub
```

Для каждого ряда на диаграмме свойствам Values, XValues и Name назначаются собственные значения. Хотя этим свойствам можно присваивать ссылки на диапазоны, свойства всегда возвращают массивы. Эта особенность обеспечивает возможность преобразования ссылки на диапазон в массив.

Помните, что количество точек данных в массиве не может превышать 250 элементов. Если это ограничение превышено, код выдаст сообщение об ошибке. На этот случай в подпрограмме предусмотрен обработчик ошибок.

Определение использованных в диаграмме диапазонов

Поведение свойств диаграммы, которое полезно для преобразовании диапазона в массив, оказывается проблемой при программном определении диапазона, на основе которого создавалась диаграмма. Если бы свойства Values и XValues возвращали строковые значения, проблема решалась бы предельно просто.

Единственным свойством, которое возвращает информацию о диапазонах, является свойство Formula. Оно возвращает функцию SERIES в виде строки. Формула может выглядеть следующим образом:

```
=SERIES(Sheet1!$A$4,Sheet1!$B$3:$D$3,Sheet1!$B$4:$D$4,1)
```

Свойство XValues определяется последними двумя параметрами, а свойство Values — третьим параметром. Для получения необходимой информации нужно найти запятые и извлечь текст, который находится между ними. Следующий код предназначен для работы с диаграммой, встроенной в активный лист:

```
Public Sub GetRangesFromChart()
    Dim aSeries As Series
    Dim seriesFunction As String
    Dim firstComma As Integer
    Dim secondComma As Integer
    Dim thirdComma As Integer
    Dim rangeValueString As String
    Dim rangeXValueString As String
    Dim rangeValue As Range
    Dim rangeXValue As Range

    On Error GoTo Catch

  ' Получить функцию SERIES для первого ряда диаграммы
  Set aSeries = ActiveSheet.ChartObjects(1).Chart.SeriesCollection(1)
  seriesFunction = aSeries.Formula

  ' Найти запятые
    firstComma = InStr(1, seriesFunction, ",")
    secondComma = InStr(firstComma + 1, seriesFunction, ",")
    thirdComma = InStr(secondComma + 1, seriesFunction, ",")
  ' Извлечь строковое представление ссылок на диапазоны
    rangeXValueString = Mid(seriesFunction, firstComma + 1, _
        secondComma - firstComma - 1)
```

```
    rangeValueString = Mid(seriesFunction, secondComma + 1, _
        thirdComma - secondComma - 1)

  ' Преобразовать строки в объекты Range
    Set rangeXValue = Range(rangeXValueString)
    Set rangeValue = Range(rangeValueString)
  ' Выделить диапазоны разными цветами
    rangeXValue.Interior.ColorIndex = 3
    rangeValue.Interior.ColorIndex = 4
    Exit Sub

Catch:
    MsgBox "Извините, произошла ошибка" & vbCr & _
       "Возможно, в диаграмме содержаться некорректные ссылки на
↳диапазоны"
End Sub
```

Переменной `seriesFunction` присваивается формула ряда. В этой формуле содержится строковое представление функции `SERIES`. С помощью функции `InStr` определяется положение первой, второй и третьей запятых. Функция `Mid` позволяет извлечь строковое представление ссылки на диапазон. Преобразование строк в объекты `Range` осуществляется с помощью свойства `Range`.

Преобразование строк в объекты `Range` выполняется даже для диапазонов из других листов и книг. Достаточно, чтобы исходные данные находились в открытой книге.

После получения диапазона можно обрабатывать объект `Range`. Например, можно программно изменить значения в ячейках диапазона или размер диапазона. В данном случае для демонстрации программного управления выполнено выделение диапазонов разным цветом.

Метки диаграмм

В Excel предоставляется возможность добавления меток данных на диаграмму. Для этого метки должны быть основаны на рядах значений или на значениях оси X. Все эти возможности доступны в диалоговом окне Диаграмма⇨Параметры диаграммы (Chart⇨Chart Options).

Кроме того, в качестве меток можно вводить собственный текст или текст формулы, ссылающейся на ячейки, но создание таких меток требует значительных трудозатрат. Для этого ряду необходимо добавить стандартные метки, после чего отдельно выделять каждую метку и редактировать текст метки или модифицировать текст формулы в поле Формула (Formula). Эти же операции могут выполняться в результате работы макроса.

Ниже показана диаграмма ежемесячного уровня продаж (рис. 24.6) каждого торгового представителя. Метки определяются формулами, связанными с четвертой строкой листа (например, "Jenny" в апреле). Формула в поле Формула (Formula) ссылается на ячейку E4.

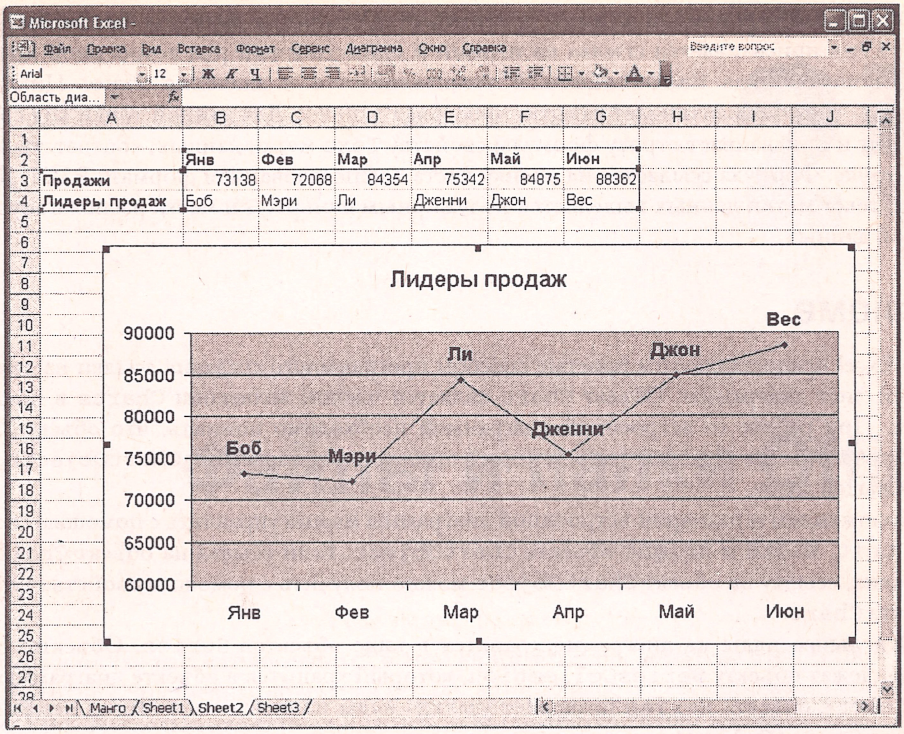


Рис. 24.6. Создание меток диаграмм

Предположим, что создана похожая небольшая диаграмма без меток данных. Следующий код позволяет добавить метки данных и ввести формулы, определяющие текст меток:

```
Public Sub AddDataLabels()
    Dim sales As Series
    Dim points As points
    Dim aPoint As Point
    Dim aRange As Range
    Dim I As Integer
    Set aRange = Range("B4:G4")
    Set sales = ActiveSheet.ChartObjects(1).Chart.SeriesCollection(1)
    sales.HasDataLabels = True
    Set points = sales.points
    For Each aPoint In points
        I = I + 1
        aPoint.DataLabel.Text = "=" & aRange.Cells(I).Address _
            (RowAbsolute:=True, _
            ColumnAbsolute:=True, _
            ReferenceStyle:=xlR1C1, _
            External:=True)
        aPoint.DataLabel.Font.Bold = True
        aPoint.DataLabel.Position = xlLabelPositionAbove
    Next aPoint
End Sub
```

Объектной переменной aRange присваивается ссылка на диапазон B4:G4. Переменной sales присваивается ссылка на первый и единственный ряд встроенной диаграммы. Свойство ряда HasDataLabels устанавливается в значение True. Цикл For Each... Next обрабатывает каждую точку ряда данных. Для каждой точки код создает формулу и присваивает строку формулы свойству Text метки данных, соответствующей этой точке. Формула ссылается на ячейку листа с использованием формата R1C1. Кроме этого, текст метки данных выделяется полужирным шрифтом и метка располагается над точкой данных.

Резюме

В Excel предоставляются простые способы программного создания ссылки на диаграмму или лист диаграммы. Объект Chart является членом коллекции Charts в пределах книги. При работе со встроенной диаграммой необходимо помнить, что объект Chart хранится в объекте ChartObject, который принадлежит коллекции ChartObjects, принадлежащей листу.

Положением и размером встроенной диаграммы можно управлять с помощью свойств Top, Left, Width и Height объекта ChartObject. Если ссылка на объект Chart уже получена, ссылку на объект ChartObject можно получить с помощью свойства Parent объекта Chart.

Отдельные ряды на диаграмме хранятся в виде объектов Series. Объекты рядов принадлежат объекту SeriesCollection, который хранится в объекте диаграммы. Метод Delete объекта Series позволяет удалять ряды на диаграмме. Метод NewSeries объекта SeriesCollection дает возможность создавать новые ряды.

Свойству Value объекта Series можно присваивать массивы VBA, а не объекты Range. При этом создается диаграмма, не зависящая от данных на листе. Такая диаграмма может распространяться без исходного листа с данными.

Свойства Values и XValues возвращают значения данных, а не ссылки на диапазоны с данными диаграммы. Для получения ссылки на диапазон с данными можно обратиться к функции SERIES, которая хранится в свойстве Formula каждого ряда.

Точка данных на диаграмме является объектом Point и принадлежит коллекции Points объекта Series. В пользовательском интерфейсе Excel не предоставляются способы связывания значения ячейки с меткой точки данных. Но такие связи могут создаваться с помощью кода VBA.

25

Файлы и папки Office

Большинство рассматривавшихся в этой книге объектов доступны в объектной модели Excel. Это не единственная объектная модель, которая используется при написании кода на Excel VBA. Несколько объектных моделей (или библиотек типов) автоматически связываются с каждым проектом Excel VBA. Список этих библиотек предоставляется в раскрывающемся списке в окне Object Browser (Просмотр объектов) или в окне, появляющемся по команде меню Tools⇨References (Сервис⇨Ссылки) в редакторе VBE. Существующие ссылки можно определить по установленному флажку напротив названия библиотеки. В списке перечислены библиотеки Excel, VBA, Office, OLE Automation и Forms (если в приложении используются диалоговые окна `UserForm`). Для подключения дополнительных библиотек необходимо установить соответствующие флажки в диалоговом окне, доступном по команде Tools⇨References (Сервис⇨Ссылки). Этот процесс рассматривался в предыдущих главах, например в главе 14.

Объектная модель Office доступна всем приложениям из состава Microsoft Office. В ней содержатся объекты, используемые во всех приложениях Office. Одним из таких объектов является `CommandBar`, который будет рассмотрен в следующей главе. В этой же главе рассматриваются два объекта, доступные в объектной модели Office и применяемые для поиска, открытия и сохранения файлов, и связанные с ними свойства:

- `FileSearch`
  - `FoundFiles`
  - `PropertyTests`
  - `FileTypes`
  - `SearchScopes`
  - `ScopeFolders`
  - `SearchFolders`

- FileDialog
  - FileDialogFilters
  - FileDialogSelectedItems

Объект FileSearch впервые появился в пакете Office 97 и с выходом Office XP был значительно улучшен. Этот объект позволяет выполнять поиск файлов по большому количеству критериев, например по типу, размеру, расположению и времени последней модификации. Объект FileSearch добавляет имена файлов в коллекцию FoundFiles.

Объект FileSearch можно использовать для операций над группами файлов вместо функции VBA Dir. Например, этот объект позволяет обнаруживать файлы определенного возраста и удалять или перемещать их в архивный каталог. Кроме этого, объект FireSearch полезен при получении данных из нескольких связанных файлов. Например, файлы, находящиеся в каталоге, имя которого начинается с имени клиента, можно консолидировать в один общий файл.

Объект FileDialog появился только в Office XP. Этот объект позволяет выводить диалоговые окна Открыть файл (Open File) и Сохранить файл как (Save File As), а также предоставляет обозреватель для поиска файлов. Объект FileDialog является более мощной версией методов GetOpenFileName и GetSaveAsFileName объекта Application, который доступен во всех версиях Excel, поддерживающих язык VBA, но недоступен в других приложениях Office. Так как объект FileDialog входит в объектную библиотеку Office, он доступен всем приложениям Office XP.

Объект FileSearch

Следующий код выполняет поиск файлов Excel в каталоге C:\Temp и всех его подкаталогах.

```
Option Explicit

Public Sub FindClientExcelFiles()
    Dim FileName As Variant
    Dim Message As String
    Dim I As Long
    Dim Count As Long

    With Application.FileSearch

        ' Подготовить критерий поиска
        .NewSearch
        .LookIn = "C:\Temp"
        .SearchSubFolders = True
        .FileType = MsoFileTypeExcelWorkbooks
        .LastModified = MsoLastModified.msoLastModifiedAnyTime
        Count = .Execute

        ' Подготовить текст сообщения
        Message = Format(Count, "###")
        Message = "Найдено файлов: " & Message
        For Each FileName In .FoundFiles
            Message = Message & vbCr & FileName
        Next FileName

        Call MsgBox(Message, vbInformation)
```

```
    End With

End Sub
```

В этом листинге для получения ссылки на объект Office.FileSearch через свойство Application.FileSearch используется конструкция With... End With. Значение свойства Application.FileSearch можно присвоить локальной переменной, но обычно это только добавляет лишние строки кода.

После этого необходимо инициализировать критерий поиска. Обычно в качестве критерия предоставляется информация о расположении файла, маска имени файла и остальные необходимые параметры. Свойство LookIn определяет подкаталог, в котором будет вестись поиск. Метод NewSearch очищает все свойства объекта FileSearch, кроме свойства LookIn. Так как эти свойства сохраняют значения, пока открыто приложение Excel, желательно явно указывать начало нового поиска с помощью метода NewSearch. Свойство SearchSubFolders управляет поиском в подкаталогах каталога, указанного в свойстве LookIn.

Свойство FileType определяет расширения файлов, которые включаются в критерий поиска. Константа msoFileTypeExcelWorkbook позволяет включить в критерий все расширения файлов Excel (.xls, .xlt, .xlm, .xlc и .xla). Ниже приводится таблица с доступными константами.

Константы msoFileType
MsoFileTypeAllFiles
MsoFileTypeBinders
MsoFileTypeCalendarItem
MsoFileTypeContactItem
MsoFileTypeCustom
MsoFileTypeDatabases
MsoFileTypeDataConnectionFiles
MsoFileTypeDesignerFiles
MsoFileTypeDocumentImagingFiles
MsoFileTypeExcelWorkbooks
MsoFileTypeJournalItem
MsoFileTypeMailItem
MsoFileTypeNoteItem
MsoFileTypeOfficeFiles
MsoFileTypeOutlookItems
MsoFileTypePhotoDrawFiles
MsoFileTypePowerPointPresentations
MsoFileTypeProjectFiles
MsoFileTypePublisherFiles
MsoFileTypeTaskItem
MsoFileTypeTemplates
MsoFileTypeVisioFiles
MsoFileTypeWebPages
MsoFileTypeWordDocuments

Свойству LastModified можно присваивать следующие константы:

Константы msoLastModified

```
MsoLastModifiedAnyTime
MsoLastModifiedLastMonth
MsoLastModifiedLastWeek
MsoLastModifiedThisMonth
MsoLastModifiedThisWeek
MsoLastModifiedToday
MsoLastModifiedYesterday
```

Вместо использования свойства FileType можно указать маску имени файла в свойстве FileName, например:

```
.FileName = "*.xls"
```

> Свойство FileName позволяет указывать более конкретный критерий для выбора имени файла, чем свойство FileType. Если одновременно воспользоваться свойствами FileType и FileName, свойство FileName переопределит свойство FileType.

Кроме этого, можно искать текст, содержащийся в свойствах файла или в теле файла. Для этого интересующий текст необходимо присвоить свойству TextOfProperty объекта FileSearch.

Свойство FoundFiles

Метод Execute объекта FileSearch выполняет поиск и добавляет в коллекцию FoundFiles объекты, представляющие найденные файлы. Кроме этого, метод Execute возвращает количество найденных файлов.

С помощью коллекции FoundFiles можно получить доступ к именам и пути найденных файлов. В показанном ранее коде для обработки списка используется цикл For Each... Next. Имя каждого файла добавляется в переменную Message. Между именами файлов вставляется символ возврата каретки. В результате работы этого кода выдается следующее диалоговое окно (рис. 25.1), фактическое содержимое которого будет отличаться в зависимости от файлов, хранящихся на компьютере.

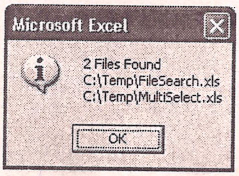


Рис. 25.1. Результат поиска файлов

Свойство PropertyTests

В Office 97 и Office 2000 в диалоговом окне Файл⇨Открыть (File⇨Open) предоставлялась кнопка Дополнительно (Advanced), щелчок на которой открывал диалоговое окно Расширенный поиск (Advanced Find). В этом диалоговом окне указывался один или больше критериев поиска файлов. Коллекцию PropertyTests можно использовать в VBA для описания условий поиска, подобных условиям в диалоговом окне Расширенный поиск (Advanced Find).

В Office XP диалоговое окно Расширенный поиск (Advanced Find) было замещено командой Файл⇨Поиск (File⇨Search), но даже в Office XP можно использовать коллекцию PropertyTests из кода VBA.

Следующая процедура ищет файлы Excel, размер которых превышает 5000 байт (новый код выделен полужирным шрифтом).

```
Public Sub FindLargeClientExcelFiles()
    Dim FileName As Variant
    Dim Message As String
    Dim I As Long
    Dim Count As Long

    With Application.FileSearch

        .NewSearch
        .LookIn = "C:\WINDOWS"
        .SearchSubFolders = True

        With .PropertyTests
            For I = .Count To 1 Step -1
                .Remove I
            Next I

            .Add Name:="Files of Type", _
            Condition:=msoConditionFileTypeExcelWorkbooks

            .Add Name:="Size", _
            Condition:=msoConditionAtLeast, _
            Value:=5000, _
            Connector:=msoConnectorAnd
        End With

        .LastModified = msoLastModifiedAnyTime
        Count = .Execute

        Message = Format(Count, "###")
        Message = "Найдено файлов: " & Message

    For Each FileName In .FoundFiles
      Message = Message & vbCr & FileName
    Next FileName

    Call MsgBox(Message, vbInformation)

  End With

End Sub
```

Коллекция PropertyTests работает независимо от параметров в диалоговом окне Расширенный поиск (Advanced Find). Она не распознает условия из этого диалогового окна и не меняет его установленные параметры. Условия в коллекции PropertyTests сохраняются до вызова метода NewSearch.

Метод Add коллекции PropertyTests добавляет новые условия. Условия указываются через присвоение строки параметру Name. Строка имеет тот же формат, который используется в раскрывающемся списке Свойство (Property) в диалоговом окне Расширенный поиск (Advanced Find). Первым добавляется условие "Файлы типа". Это еще один способ указать, что необходимо искать файлы Excel. Для этого параметру Condition присваивается соответствующая константа. Список доступных констант слишком велик, чтобы приводить его здесь. Технология InterlliSense обеспечивает автоматический ввод констант при редактировании кода, а названия констант говорят сами за себя.

Неприятной особенностью коллекции PropertyTests является принятое по умолчанию условие, которое определяется при запуске Excel и после вызова метода NewSearch. В этом условии параметр "Файлы типа" определяется как msoFileTypeOfficeFiles. Таким образом, желательно очистить коллекцию до ввода новых условий, что позволит начать поиск без дополнительных условий. Код в цикле For... Next удаляет элементы коллекции в обратном порядке. Сначала удаляются элементы с наибольшим порядковым номером. В противном случае при каждом удалении элемента менялись бы порядковые номера элементов коллекции. В качестве второго условия указывается параметр "Размер", имеющий значение 5000 байт. Параметр Connector имеет значение msoConnectorAnd. Это значение параметра указывает соединение первого условия со вторым с помощью логического оператора And. Кроме этого, можно использовать значение msoConnectorOr. Полный список доступных значений параметра приводится в приложении В.

> Если при использовании Office 97 или Office 2000 вместо значения msoConnectorAnd указать значение msoConnectorOr, интерпретатор VBA аварийно завершит работу с сообщением об ошибке. Как обычно, в этом сообщении не содержится полезной информации, позволяющей определить причину проблемы. А проблема в том, что условие "Файлы типа" нельзя комбинировать с другими условиями с помощью логического оператора Or. Если попытаться это сделать вручную в диалоговом окне Расширенный поиск (Advanced Find), будет выдано понятное сообщение о невозможности такой комбинации. Если нарушить это правило в Office XP, ошибка не возникнет, но для получения результата необходимо использовать оператор And. Помните об этом.

Коллекция FileTypes

В Office 97 и Office 2000 можно указать только одно значение свойства FileType объекта FileSearch. В Office XP предоставляется коллекция FileTypes, позволяющая определить несколько условий для типов файлов. Следующий код для Office XP находит все файлы Excel и Word в указанных каталогах:

```
Public Sub FindClientExcelAndWordFiles()
    Dim FileName As Variant
    Dim Message As String
    Dim I As Long
    Dim Count As Long

    With Application.FileSearch
        .NewSearch
```

```
        .FileType = msoFileTypeExcelWorkbooks
        .FileTypes.Add msoFileTypeWordDocuments
        .LookIn = "C:\WINDOWS"

        .SearchSubFolders = True
        .LastModified = msoLastModifiedAnyTime
        Count = .Execute

        Message = Format(Count, "###")
        Message = "Найдено файлов: " & Message

        For Each FileName In .FoundFiles
            Message = Message & vbCr & FileName
        Next FileName

        Call MsgBox(Message, vbInformation)

    End With

End Sub
```

Коллекция FileTypes сохраняется до вызова метода NewSearch. Этот метод очищает коллекцию и добавляет значение msoFileTypeOfficeFiles, но очистка коллекции перед добавлением новых записей не требуется. Если свойству FileType присвоить значение, все существующие записи в коллекции FileTypes удаляются и значение свойства FileType становится единственным элементом коллекции FileTypes. После этого для добавления элементов коллекции можно использовать метод Add. Для этого применяются те же константы типов файлов, которые были показаны ранее.

Коллекция SearchScopes

Во всех показанных ранее фрагментах кода предполагается, что структура каталогов на компьютере известна и разработчик может указать список каталогов, в которых необходимо искать интересующие файлы. Но что, если структура неизвестна и ее необходимо выяснить? Эта функциональность может потребоваться каждому приложению, которое должно работать на любом компьютере.

Коллекция SearchScopes впервые появилась в Office XP и предоставила механизм для выяснения структуры каталогов. Следующий код рассматривает каждый элемент коллекции, представляющий собой объект SearchScope.

```
Public Sub ListSearchScopeObjects()
    Dim SearchScope As SearchScope
    Dim Message As String

    For Each SearchScope In Application.FileSearch.SearchScopes
        Message = Message & SearchScope.ScopeFolder.Name & vbTab
        Message = Message & " Type=" & SearchScope.Type & vbCr
    Next SearchScope

    Call MsgBox(Message, vbInformation)

End Sub
```

Этот код выдаст следующее окно сообщения (рис. 25.2).

Объекты SearchScope представляют рассматриваемую структуру. Свойство Type позволяет идентифицировать категорию каждой структуры. Совершенно не удивляет присутствие папок Мой компьютер (My Computer) и Сетевое окружение (Network Places).

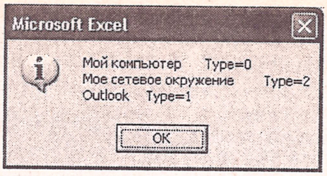


Рис. 25.2. Области поиска файлов

В справочном руководстве приводятся следующие константы, представляющие возможные значения типов:

Константы msoSearchIn
MsoSearchInCustom
MsoSearchInMyComputer
MsoSearchInMyNetworkPlaces
MsoSearchInOutlook

Добавление и удаление членов коллекции SearchScopes не поддерживается. Скорее всего, тип Custom будет использоваться в будущих разработках.

Свойство ScopeFolder

Каждый объект SearchScope имеет свойство ScopeFolder, в котором содержится ссылка на объект ScopeFolder. Объект ScopeFolder описывает верхний уровень структуры. В показанном ранее коде свойство Name объекта ScopeFolder содержит описание соответствующей структуры.

Внутри объекта ScopeFolder, представляющего верхний уровень структуры, хранится коллекция ScopeFolders, содержащая дополнительные объекты ScopeFolder. Следующий код выводит значения свойств Name и Path объектов ScopeFolder под объектом ScopeFolder каждой структуры:

```
Public Sub ListScopeFolderObjects()
  Dim SearchScope As SearchScope
  Dim ScopeFolder As ScopeFolder
  Dim Message As String

  Application.FileSearch.RefreshScopes

  For Each SearchScope In Application.FileSearch.SearchScopes
    Select Case SearchScope.Type

      Case msoSearchInMyComputer
        Message = SearchScope.ScopeFolder.Name & vbCr
        For Each ScopeFolder In SearchScope.ScopeFolder.ScopeFolders
          Message = Message & ScopeFolder.Name & vbTab & vbTab
          Message = Message & "Path=" & ScopeFolder.Path & vbCr
        Next ScopeFolder
      Case msoSearchInMyNetworkPlaces
        Message = Message & vbCr & SearchScope.ScopeFolder.Name & vbCr
        For Each ScopeFolder In SearchScope.ScopeFolder.ScopeFolders
           Message = Message & ScopeFolder.Name & vbTab
           Message = Message & "Path=" & ScopeFolder.Path & vbCr
      Next ScopeFolder
```

```
        Case msoSearchInOutlook
      Message = Message & vbCr & SearchScope.ScopeFolder.Name & vbCr
      For Each ScopeFolder In SearchScope.ScopeFolder.ScopeFolders
        Message = Message & ScopeFolder.Name & vbTab & vbTab
        Message = Message & "Path=" & ScopeFolder.Path & vbCr
      Next ScopeFolder
              Case Else
      Message = Message & vbCr & "Неизвестный объект SearchScope"
      End Select
  Next SearchScope
   Call MsgBox(Message, vbInformation)
End Sub
```

В результате работы кода (при отсутствии подключения к сети) выдается следующее диалоговое окно (рис. 25.3).

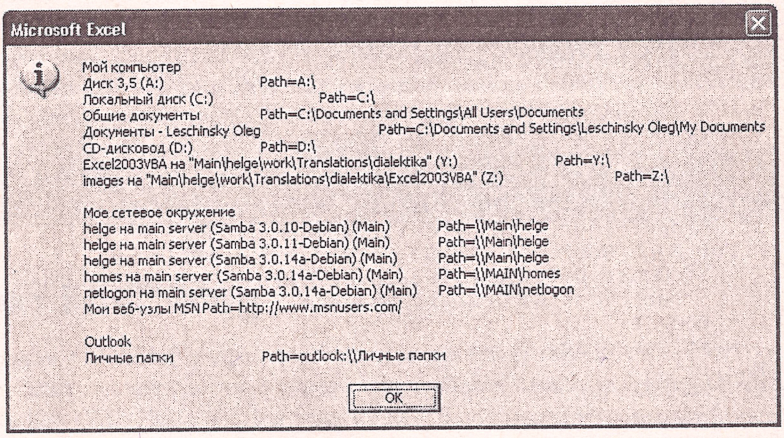


Рис. 25.3. Структура каталогов, в которых будет выполняться поиск файлов

Оператор `Select Case` предоставляет удобный способ изоляции и проверки каждой структуры верхнего уровня. Любая коллекция `ScopeFolders` верхнего уровня содержит объекты `ScopeFolder`, представляющие корневые каталоги доступных файловых структур. Каждый объект `ScopeFolder` содержит собственные коллекции `ScopeFolders` для описания собственных подкаталогов. Такая структура предоставляет механизм спуска по дереву каталогов.

Коллекция SearchFolders

Коллекция `SearchFolders` также появилась в объектной модели с выходом Office XP. Свойство `LookIn` объекта `FileSearch` определяет единственный каталог, в котором будет выполняться поиск. Свойству присваивается строковое представление пути к каталогу. Для определения дополнительных каталогов, в которых будет выполняться поиск, используется коллекция `SearchFolders`. Для этого в коллекцию необходимо добавить новые объекты `ScopeFolder`.

В отличие от коллекции `FileTypes`, создаваемой заново при установке свойства `FileType`, коллекция `SearchFolders` не меняется при установке свойства `LookIn` и при вызове метода `NewSearch`. Значение свойства `LookIn` используется совместно с элементами коллекции `SearchFolders`.

Так как коллекция SearchFolders сохраняет состояние на протяжении всего сеанса работы с Excel, желательно очищать ее перед началом нового поиска. Следующий код выполняет поиск во всех подкаталогах корневого каталога диска С. При обнаружении каталога, имя которого содержит строку "System32", этот каталог добавляется в коллекцию SearchFolders.

```
Public Sub SetupSearchFoldersCollection()
  Dim FileSearch As FileSearch
  Dim SearchScope As SearchScope
  Dim ScopeFolder As ScopeFolder
  Dim SubFolder As ScopeFolder
  Dim Message As String
  Dim I As Long

  Set FileSearch = Application.FileSearch

  For I = FileSearch.SearchFolders.Count To 1 Step -1
    FileSearch.SearchFolders.Remove I
  Next I

  For Each SearchScope In FileSearch.SearchScopes
    Select Case SearchScope.Type
      Case msoSearchInMyComputer
        For Each ScopeFolder In SearchScope.ScopeFolder.ScopeFolders
          If (ScopeFolder.Path = "C:\") Then
            For Each SubFolder In ScopeFolder.ScopeFolders
              If UCase(Left(SubFolder.Name, 6)) = "SYSTEM32" Then
                SubFolder.AddToSearchFolders
              End If
            Next SubFolder
          End If
        Next ScopeFolder
    End Select
    Next SearchScope

  PerformSearch

End Sub
```

Код очищает коллекцию SearchFolders и просматривает коллекцию SearchScopes и коллекцию ScopeFolders, пока не обнаружит диск С. После этого просматривается свойство Name каждого объекта ScopeFolder из корневого каталога диска С. При этом код пытается найти каталоги, имя которых начинается со строки "SYSTEM32". Для обеспечения нечувствительного к регистру сравнения имена каталогов преобразовываются в верхний регистр.

После обнаружения подходящего каталога объект добавляется в коллекцию SearchFolders с помощью метода AddToSearchFolders объекта ScopeFolder. После этого код вызывает подпрограмму PerformSearch, которая показана ниже. Эта подпрограмма выводит имена файлов Excel, расположенных в каталоге SYSTEM32:

```
Public Sub PerformSearch()
  Dim FileName As Variant
  Dim Message As String
  Dim Count As Long

  With Application.FileSearch
    .NewSearch
    .LookIn = ""
```

```
    .SearchSubFolders = True
    .FileName = "*.xls"
    .LastModified = msoLastModifiedAnyTime
    Count = .Execute

        Message = Format(Count, "###")
        Message = "Найдено файлов: " & Message

    For Each FileName In .FoundFiles
      Message = Message & vbCr & FileName
    Next FileName
  End With

  MsgBox Message

End Sub
```

Подпрограмма PerformSearch устанавливает свойство LookIn объекта FileSearch равным строке нулевой длины, что позволяет избавиться от ссылок на каталоги, оставшихся после предыдущего использования объекта FileSearch.

> Пользователям Office XP придется модифицировать код показанных ранее процедур таким образом, чтобы коллекция SearchFolders очищалась перед выполнением поиска.

Объект FileDialog

С выходом Office XP разработчикам стал доступен объект FileDialog, позволяющий коду VBA выводить диалоговые окна Файл⇨Открыть (File⇨Open) и Файл⇨Сохранить Как (File⇨SaveAs). Для этих же целей пользователи предыдущих версий Excel могут применить методы GetOpenFileName и GetSaveAsFileName объекта Application. Обратная совместимость с этой возможностью сохраняется и в Office 2003. Одним из преимуществ объекта FileDialog является дополнительная возможность просмотра отдельного списка каталогов. Кроме этого, объект FileDialog доступен во всех приложениях Office.

В данном примере создается лист, на котором выводятся изображения, выбранные пользователем в диалоговом окне Файл⇨Открыть (File⇨Open) (рис. 25.4).

На листе содержится элемент управления Рисунок (Image), доступный на панели инструментов Элементы управления (Control Toolbox). Элементу управления присвоено имя “Image1”. Для автоматического изменения размеров изображения в соответствии с размерами элемента управления используется свойство PictureSizeMode (свойство устанавливается в значение zoom). Расположенная выше кнопка имеет принятое по умолчанию имя и надпись Select File. Модуль класса листа Sheet1 содержит следующую процедуру обработки события:

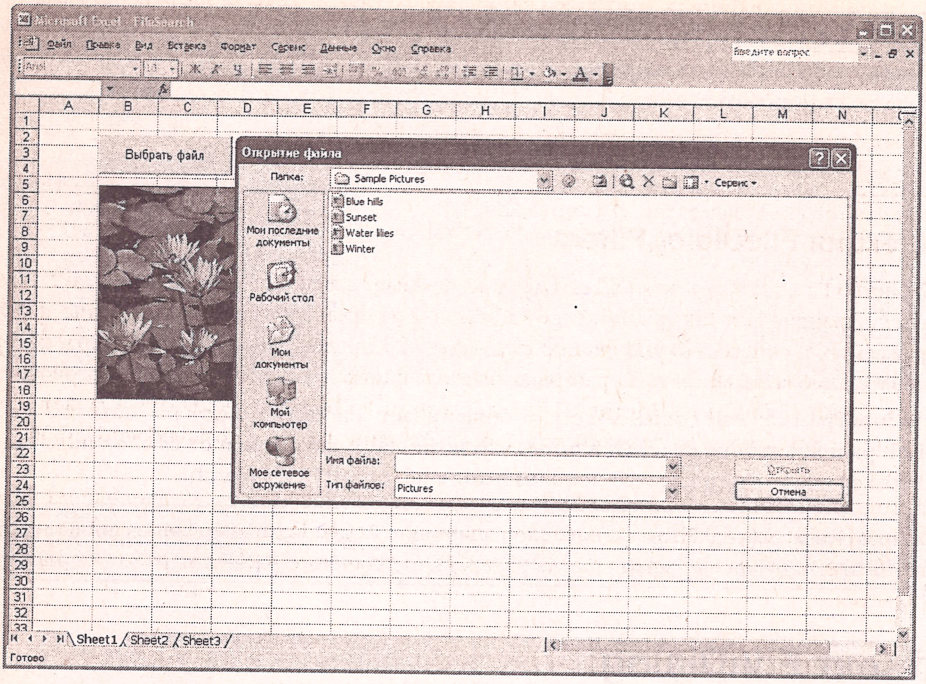


Рис. 25.4. Выбор изображения с помощью диалогового окна Открыть файл

```
Private Sub CommandButton1_Click()
  Dim Filters As FileDialogFilters
  Dim FileName As String

  On Error GoTo Catch

  With Application.FileDialog(MsoFileDialogType.msoFileDialogOpen)
    Set Filters = .Filters

    With Filters
      .Clear
      .Add "Pictures", "*.jpg"
    End With

    .AllowMultiSelect = False

    If .Show = False Then Exit Sub

    Image1.Picture = LoadPicture(.SelectedItems(1))
  End With
  Exit Sub

Catch:
  Call MsgBox("Некорректный файл изображения", vbExclamation)
End Sub
```

Свойство `FileDialog` объекта `Application` возвращает ссылку на объект `FileDialogs`. Тип диалогового окна определяется следующими константами `msoFileDialogType`:

Константы MsoFileDialog
`MsoFileDialogFilePicker`
`MsoFileDialogFolderPicker`
`MsoFileDialogOpen`
`MsoFileDialogSaveAs`

Коллекция FileDialogFilters

Свойство `Filters` объекта `FileDialog` возвращает ссылку на коллекцию `FileDialogFilters`, хранящуюся внутри объекта `FileDialog`. Фильтры определяют типы файлов, отображаемых в списке. По умолчанию существует 24 предопределенных фильтра, которые пользователь может выбирать из раскрывающегося списка в нижней части диалогового окна Открыть файл (File Open). Метод `Clear` коллекции `FileDialogFilters` удаляет предопределенные фильтры. После этого код добавляет один фильтр, позволяющий просматривать список файлов `.jpg`.

Метод `Show` объекта `FileDialog` выводит диалоговое окно. При щелчке на кнопке Открыть (Open) метод `Show` возвращает значение `True`. Если пользователь щелкает на кнопке Отмена (Cancel), метод `Show` возвращает значение `False` и работа процедуры завершается.

Коллекция FileDialogSelectedItems

На самом деле метод `Show` не открывает выбранный файл. Вместо этого имя и путь выбранного файла добавляются в коллекцию `FileDialogSelectedItems`. Как будет показано ниже, в диалоговом окне можно выбрать несколько файлов. В предыдущем примере пользователь мог выбирать только один файл. Имя файла извлекается из первого элемента коллекции `FileDialogSelectedItems`, на которую ссылается свойство `SelectedItems` объекта `FileDialog`.

Функция `LoadPicture` присваивает файл свойству `Picture` элемента управления `Image`.

Типы диалоговых окон

Диалоговые окна различных типов практически ничем не отличаются, кроме заголовка. В диалоговых окнах для выбора файлов и папок в заголовке отображается строка Поиск (Browse), а в остальных типах – строка Открыть файл (File Open) или Сохранить файл как (File Save As). Диалоговые окна всех типов, кроме окна выбора папок, отображают список файлов и каталогов. В окне выбора папок отображается только список каталогов.

Метод Execute

Как было показано ранее, метод `Show` выводит диалоговое окно `FileDialog` и выбранные в диалоговом окне элементы добавляются в коллекцию `FileDialogSelectedItems`. Диалоговое окно не пытается открыть или сохранить файлы. Метод `Execute` заставляет диалоговое окно выполнить необходимую операцию открытия или сохранения немедленно после щелчка на кнопке Открыть (Open) или Сохранить (Save), например:

```
With Application.FileDialog(xlDialogOpen)
    If .Show Then .Execute
End With
```

Свойство MultiSelect

Показанное на рис. 25.5 приложение модифицировано для поддержки выбора нескольких файлов. Для этого необходимо удерживать клавиши <Shift> или <Ctrl> при щелчке на интересующих именах файлов. Выбранные имена файлов загружаются в раскрывающийся список ComboBox1 в верхней части экрана. После этого раскрывающийся список можно использовать для выбора и просмотра интересующего файла.

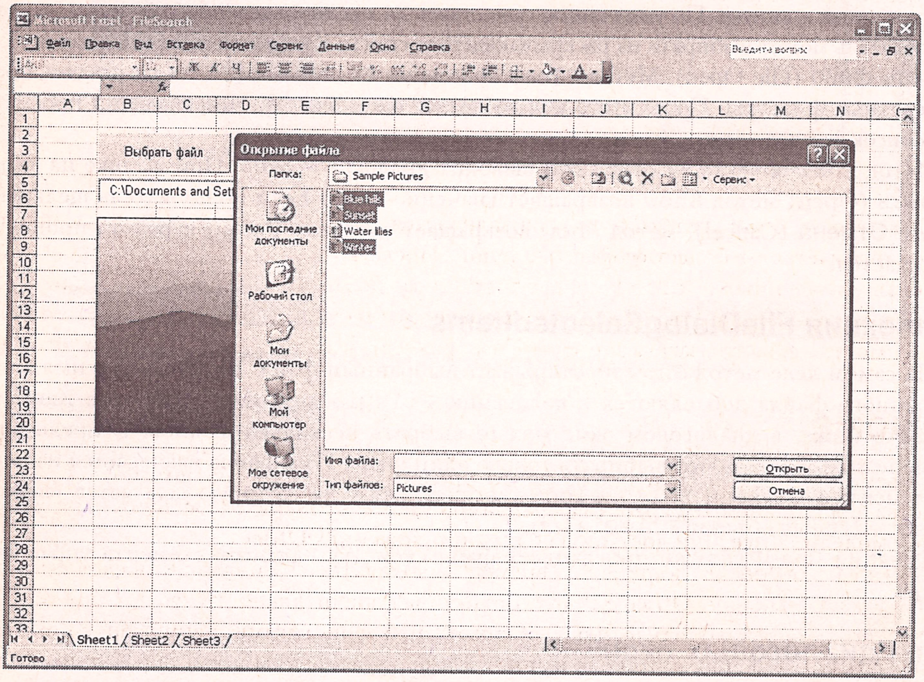


Рис. 25.5. Выбор нескольких файлов из списка

В код были внесены следующие модификации:

```
Option Explicit

Private Sub ComboBox1_Change()
  Image1.Picture = LoadPicture(ComboBox1.Text)
End Sub
Private Sub CommandButton1_Click()
  Dim FileName As String
  Dim Item As Variant

  On Error GoTo Catch

  With Application.FileDialog(msoFileDialogOpen)

    With .Filters
```

```
      .Clear
      .Add "Pictures", "*.jpg"
    End With

    .AllowMultiSelect = True

    If .Show = False Then Exit Sub
    ComboBox1.Clear

    For Each Item In .SelectedItems
      Call ComboBox1.AddItem(Item)
    Next Item

    ComboBox1.ListIndex = 0

  End With

  Exit Sub

Catch:
  Call MsgBox(Err.Description, vbExclamation)
End Sub
```

Свойство `AllowMultiSelect` устанавливается в значение `True`. Раскрывающийся список очищается от существующих значений. После этого в цикле `For Each... Next` элементы коллекции `FileDialogSelectedItems` добавляются в раскрывающийся список. При установке свойства `ListIndex` в значение 0 возникает событие `Change` и процедура обработки события загружает первое изображение в элемент управления `Image`.

Резюме

Объекты `FileSearch` и `FileDialog` предоставляют разработчикам VBA удобные средства для создания кода работы с файлами. Так как эти объекты являются частью объектной модели Office, они доступны во всех приложениях Office.

Объект `FileSearch` доступен со времен выхода Office 97 и во всех последних версиях Windows. С выходом Office XP этот объект был значительно улучшен. Объект `FileSearch` позволяет находить файлы с общими характеристиками, например, с похожими именами или одинаковым расположением. Найденные файлы могут обрабатываться последующим кодом.

26

Командные панели

Коллекция CommandBars является членом объектной модели Office, рассматриваемой в приложении В. Эта коллекция содержит все меню, панели инструментов и контекстные меню, уже встроенные в Excel и другие приложения Office, а также те, которые создаются разработчиком. Для доступа к командным панелям используется свойство CommandBars объекта Application.

Впервые командные панели появились в Office 97. В Excel 5 и Excel 95 панели меню и панели инструментов были разными типами объектов. Контекстные или всплывающие меню, появляющиеся при щелчке правой кнопкой мыши на ячейке листа, были специальным типом панели меню. В Excel 97 и в более поздних версиях “командные панели” стали универсальным термином для описания панелей меню, панелей инструментов и контекстных меню.

Командные панели содержат *элементы управления (controls)*. При щелчке некоторые элементы управления выполняют операции, например Копирование (Copy). Эти элементы до более подробного их рассмотрения назовем *командами (commands)*. Существуют и другие элементы управления (например, Файл (File)), которые при щелчке на них создают новый набор элементов управления. Такие элементы управления называются *меню (menu)*.

В данной главе рассматривается создание и управление этими инструментами.

Панели инструментов, панели меню и всплывающие меню

Ниже показано окно Excel, в верхней части которого расположена стандартная панель меню Worksheet (рис. 26.1).

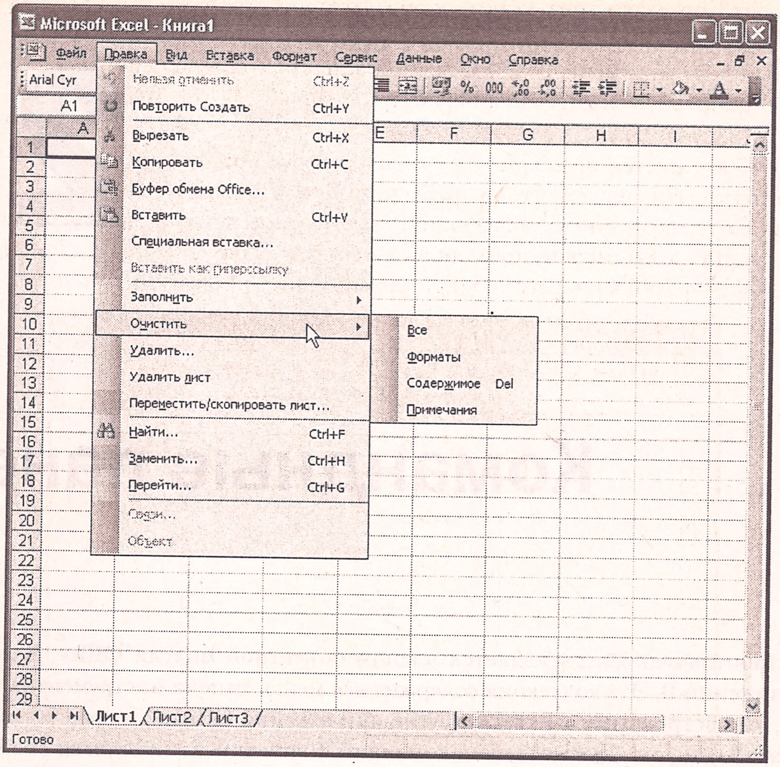


Рис. 26.1. Панель меню Worksheet

В панели меню Worksheet содержатся такие меню, как Файл (File) и Правка (Edit). При щелчке на меню выводится дополнительный список команд и меню.

- Пункты Вырезать (Cut) и Копировать (Copy) являются командами и расположены в меню Правка (Edit).
- Пункт Очистить (Clear) является примером меню внутри меню Правка (Edit).

На рис. 26.2 показана панель инструментов Стандартная (Standard).

В панелях инструментов хранятся элементы управления, которые запускают команды Excel. Например, кнопка с ножницами вызывает команду Вырезать (Cut). Кроме этого, в панелях инструментов хранятся элементы управления других типов, например раскрывающийся список Масштаб (Zoom) (второй с конца панели инструментов Стандартная (Standard) на рис. 26.2), позволяющий выбрать или ввести коэффициент масштабирования (в процентах). На некоторых панелях инструментов содержатся кнопки, щелчок на которых приводит к выводу меню, например кнопка Сводная таблица (PivotTable) на панели инструментов Сводная таблица (PivotTable).

Рис. 26.2. Панель инструментов Стандартная

На рис. 26.3 показано контекстное меню, которое отображается при щелчке правой кнопкой мыши на ячейке листа.

В этом контекстном меню содержатся команды, например Вставить (Paste), и меню, например Удалить (Delete), для операций, которые допустимы в данном контексте (в данном случае, в ячейке).

В Excel 2003 имеется примерно 124 разных встроенных командных панелей, содержащих несколько тысяч элементов управления. Кроме этого, можно создавать собственные командные панели или использовать существующие в соответствии с текущими потребностями. Для этого можно применить команду меню Вид⇨Панели инструментов⇨Настройка (View⇨Toolbars⇨Customize) или выполнить программную настройку.

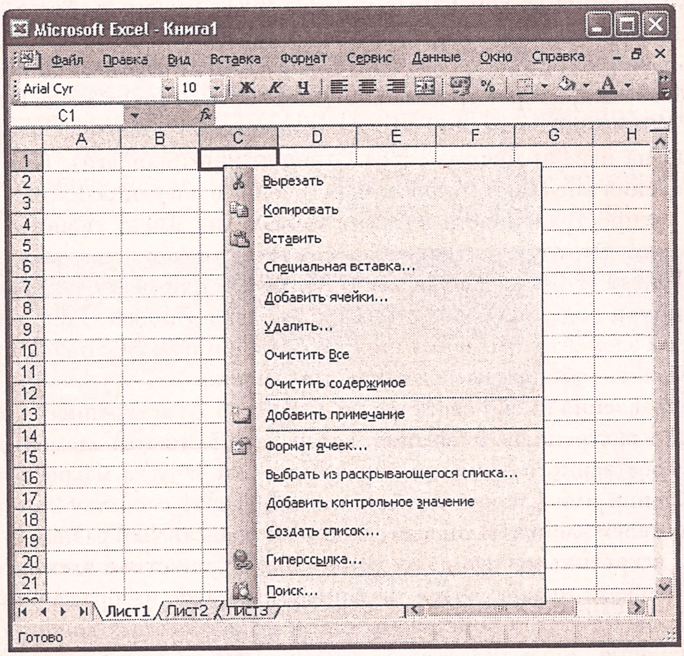


Рис. 26.3. Контекстное меню ячейки листа

Практически всю настройку можно выполнить вручную, включая связывание командных панелей с книгами, но некоторые операции выполняются только с помощью кода VBA. Например, VBA необходим для следующих операций:

- автоматическое удаление или сокрытие командной панели при закрытии или деактивизации соответствующей книги;
- добавление собственного меню во встроенную панель меню при открытии книги и удаление меню при закрытии книги;
- динамическое изменение командных панелей в ответ на действия пользователя;
- использование некоторых элементов управления, которые могут создаваться и модифицироваться только из кода VBA.

Другими словами, командные панели могут быть трех типов, а именно: меню, панель инструментов или контекстное меню. При создании командной панели с помощью VBA необходимо указать тип создаваемой панели. Тип указывается в качестве параметра метода

Add коллекции CommandBars. Примеры таких параметров приводятся в следующей таблице. Тип существующей командной панели определяется с помощью свойства Type, которое возвращает числовое значение, равное одной из следующих встроенных констант.

Константа	*Тип командной панели*
MsoBarTypeNormal	Панель инструментов
MsoBarTypeMenuBar	Панель меню
MsoBarTypePopup	Контекстное меню

Элементы управления на командных панелях также имеют свойство Type, содержащее одну из констант msoXXX. Чаще всего используется элемент управления, свойство Type которого равно msoControlButton. Такой элемент управления используется для предоставления команд, например команды Копировать (Copy) в меню Правка (Edit) на панели меню Worksheet, или кнопки на панели инструментов, например кнопки Вырезать (Cut) на панели инструментов Стандартная (Standard). При щелчке такие элементы управления запускают макрос или встроенную операцию Excel.

Второй по распространенности элемент управления имеет свойство Type, равное msoControlPopup. Этот элемент управления соответствует меню на панели меню, например меню Правка (Edit) на панели меню Worksheet или меню, содержащемуся в другом меню, например подменю Очистить (Clear) в меню Правка (Edit) на панели меню Worksheet. Такой элемент управления имеет собственную коллекцию Controls, в которую можно добавлять собственные элементы управления.

Элементы управления имеют свойство Id. Свойство Id встроенных элементов управления определяет внутреннюю операцию, которая выполняется элементом управления. При создании собственного элемента управления для запуска макроса при щелчке на элементе управления имя макроса необходимо присвоить свойству OnAction. Все созданные пользователем элементы управления имеют свойство Id, равное 1.

Со многими встроенными пунктами меню и с большинством элементов управления на встроенных панелях инструментов связано графическое изображение, которое определяется свойством FaceId. Свойства FaceId и Id встроенных команд обычно имеют одинаковые значения. Если известно конкретное числовое представление, значение свойства FaceId встроенного элемента управления можно использовать при создании собственного элемента управления. Для определения числового значения можно воспользоваться VBA, как показано в следующем примере.

Встроенные командные панели Excel

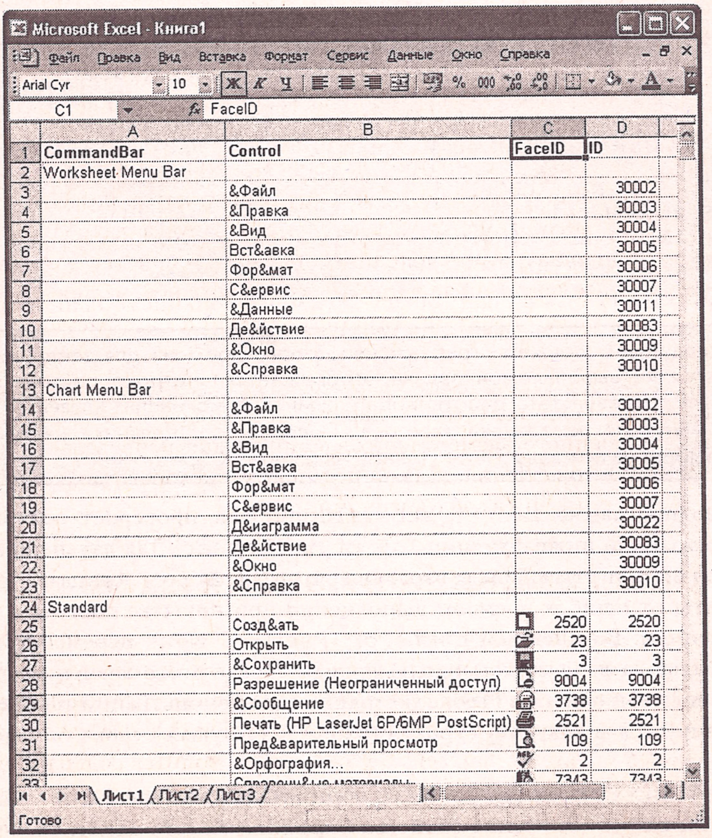
Перед началом создания собственных командных панелей желательно определить, как структурированы встроенные командные панели и какие панели уже доступны в Excel 2003. Следующий код можно использовать для просмотра существующих и добавленных пользователем командных панелей. В результате выполнения этого кода в столбце A выводится список всех командных панелей. В столбце B выводится содержимое коллекции Controls соответствующей командной панели (рис. 26.4). Этот код не пытается выводить элементы управления более низкого уровня, принадлежащие таким элементам управления, как меню Файл (File) в панели меню Worksheet, поэтому процедура называется ListFirstLevelControls.

Кроме этого, макрос выводит значение свойства `Id` каждого элемента управления. Если для элемента управления существует графическое изображение, то оно появляется вместе со значением свойства `FaceId`. Обратите внимание, что некоторые показанные в списке элементы управления могут не отображаться на экране. Например, при отсутствии почтовой системы не будет показана панель инструментов Стандартная (Standard) и кнопка Mail Recipient.

> Запускайте этот макрос и следующие два примера только в пустом листе. В этих примерах используются проверки, позволяющие защитить существующие данные от перезаписи.

Для тестирования кода добавьте подпрограмму в стандартный модуль (не модуль класса). Не добавляйте код в модуль `ThisWorkbook` или модуль класса листа. Кроме этого, необходимо ввести показанную ниже функцию `IsEmptyWorksheet`.

В приведенном примере обращает на себя внимание то, что практически во всех именах элементов управления присутствует символ амперсанда. Амперсанд используется при разработке элемента управления для определения клавиши быстрого доступа к пункту меню. Например, меню `&File` будет отображаться как File, но комбинация клавиш <Alt+F> может применяться для быстрого доступа к этому пункту меню.



	A	B	C	D
1	CommandBar	Control	FaceID	ID
2	Worksheet Menu Bar			
3		&Файл		30002
4		&Правка		30003
5		&Вид		30004
6		Вст&авка		30005
7		Фор&мат		30006
8		С&ервис		30007
9		&Данные		30011
10		Де&йствие		30083
11		&Окно		30009
12		&Справка		30010
13	Chart Menu Bar			
14		&Файл		30002
15		&Правка		30003
16		&Вид		30004
17		Вст&авка		30005
18		Фор&мат		30006
19		С&ервис		30007
20		Д&иаграмма		30022
21		Де&йствие		30083
22		&Окно		30009
23		&Справка		30010
24	Standard			
25		Созд&ать	2520	2520
26		Открыть	23	23
27		&Сохранить	3	3
28		Разрешение (Неограниченный доступ)	9004	9004
29		&Сообщение	3738	3738
30		Печать (HP LaserJet 6P/6MP PostScript)	2521	2521
31		Пред&варительный просмотр	109	109
32		&Орфография...	2	2
33		Справочн&ые материалы...	7343	7343

Рис. 26.4. Вывод списка командных панелей и элементов управления

Показанный ниже код позволяет просматривать список элементов управления первого уровня:

```
Sub ListFirstLevelControls()
    Dim cbCtl As CommandBarControl
    Dim cbBar As CommandBar
    Dim i As Integer
    If Not IsEmptyWorksheet(ActiveSheet) Then Exit Sub
        On Error Resume Next
        Application.ScreenUpdating = False
        Cells(1, 1).Value = "Командная панель"
        Cells(1, 2).Value = "Элемент управления"
        Cells(1, 3).Value = "FaceID"
        Cells(1, 4).Value = "ID"
        Cells(1, 1).Resize(1,4).Font.Bold = True
        i = 2
           For Each cbBar In CommandBars
               Application.StatusBar = "Обрабатывается панель " & _
                                       cbBar.Name
               Cells(i, 1).Value = cbBar.Name
               i = i + 1
                   For Each cbCtl In cbBar.Controls
                       Cells(i, 2).Value = cbCtl.Caption
                       cbCtl.CopyFace
                       If Err.Number = 0 Then
                           ActiveSheet.Paste Cells(i, 3)
                           Cells(i, 3).Value = cbCtl.FaceID
                       End If
                       Cells(i, 4).Value = cbCtl.ID
                       Err.Clear
                       i = i + 1
                   Next cbCtl
           Next cbBar
       Range("A:B").EntireColumn.AutoFit
    Application.StatusBar = False
End Sub
```

> Этот и два следующих примера требуют некоторого времени для завершения работы. Текущее состояние выполняемой задачи отображается в строке состояния. Для просмотра частичного результата работы кода нажмите комбинацию клавиш <Ctrl+Break> через пару минут после запуска макроса. После этого щелкните на кнопке Debug (Отладка) и выберите команду Run⇨Reset (Выполнить⇨Сброс).

Сначала подпрограмма ListFirstLevelControls проверяет содержимое листа с помощью показанной ниже функции IsEmptyWorksheet. Для дальнейшей работы кода лист должен быть пустым. После этого оператор On Error Resume Next используется для защиты от ошибок времени выполнения при попытке доступа к несуществующим изображениям. Во внешнем цикле For Each... Next переменной cbBar присваивается ссылка на каждую командную панель. После этого значение свойства Name командной панели выводится в строке состояния (для демонстрации текущего состояния задачи) и копируется в текущую строку столбца A (текущая строка определяется переменной-счетчиком i).

Внутренний цикл For Each... Next обрабатывает все элементы управления в командной панели cbBar. Значение свойства Caption каждого элемента управления выводится в столбце B. После этого код пытается использовать метод CopyFace элемента управления для копирования изображения элемента управления в буфер обмена. Если

операция копирования не приводит к появлению ошибки, изображение вставляется в столбец С и в этой же ячейке выводится значение свойства FaceId. Значение свойства Id элемента управления выводится в столбце D. После этого код удаляет все сообщения об ошибках, увеличивает значение переменной i на единицу и переходит к обработке следующего элемента управления.

Показанная ниже функция IsEmptyWorksheet проверяет, является ли предоставленный в качестве параметра объект Sht на самом деле листом. Если это так, функция проверяет, равно ли 0 количество ячеек в использованном диапазоне. Если оба условия выполняются, функция возвращает значение True. В противном случае функция выдает предупреждение и возвращает принятое по умолчанию значение False.

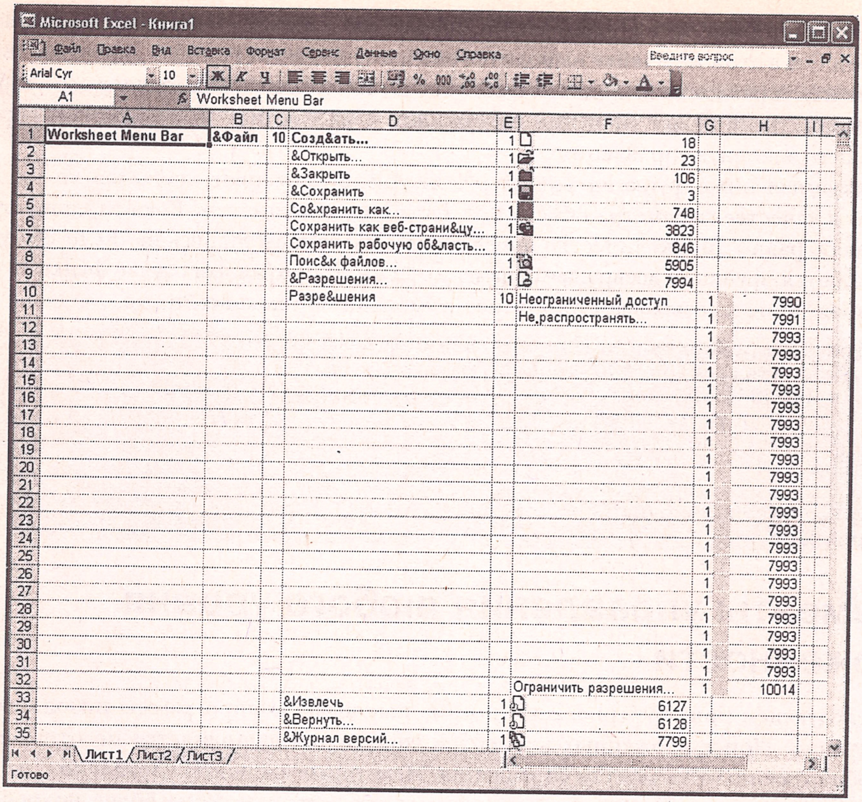
```
Function IsEmptyWorksheet(Sht As Object) As Boolean
   If TypeName(Sht) = "Worksheet" Then
      If WorksheetFunction.CountA(Sht.UsedRange) = 0 Then
         IsEmptyWorksheet = True
         Exit Function
      End If
   End If
   MsgBox "Активизируйте пустой лист"
End Function
```

Элементы управления любого уровня вложенности

В следующем примере (рис. 26.5) показанные выше подпрограммы используются более активно. Поиск вложенных элементов управления выполняется по отношению ко всем обнаруженным элементам управления. Некоторые элементы управления, например, содержащие графические изображения, не могут быть показаны более подробно. Информация о вложенных элементах управления приводится со смещением вправо. Код может выводить информацию об элементах управления на всех существующих уровнях вложенности, но Excel не использует элементы управления с уровнем вложенности больше трех.

Следующий код позволяет просматривать список элементов управления на всех уровнях вложенности:

```
Sub ListAllControls()
    Dim cbBar As CommandBar
    Dim Rng As Range
    Dim cbCtl As CommandBarControl

    If Not IsEmptyWorksheet(ActiveSheet) Then Exit Sub
    Application.ScreenUpdating = False
    Set Rng = Range("A1")
    For Each cbBar In Application.CommandBars
        Application.StatusBar = "Обрабатывается панель " & cbBar.Name
        Rng.Value = cbBar.Name
        For Each cbCtl In cbBar.Controls
            Set Rng = Rng.Offset(ListControls(cbCtl, Rng))
        Next cbCtl
    Next cbBar
    Range("A:I").EntireColumn.AutoFit
    Application.StatusBar = False
End Sub
```



	A	B	C	D	E	F	G	H
1	Worksheet Menu Bar	&Файл	10	Созд&ать...	1	18		
2				&Открыть...	1	23		
3				&Закрыть	1	106		
4				&Сохранить	1	3		
5				Со&хранить как...	1	748		
6				Сохранить как веб-страни&цу...	1	3823		
7				Сохранить рабочую об&ласть...	1	846		
8				Поис&к файлов...	1	5905		
9				&Разрешения...	1	7994		
10				Разре&шения	10	Неограниченный доступ	1	7990
11						Не,распространять...	1	7991
12							1	7993
13							1	7993
14							1	7993
15							1	7993
16							1	7993
17							1	7993
18							1	7993
19							1	7993
20							1	7993
21							1	7993
22							1	7993
23							1	7993
24							1	7993
25							1	7993
26							1	7993
27							1	7993
28							1	7993
29							1	7993
30							1	7993
31							1	7993
32						Ограничить разрешения...	1	10014
33				&Извлечь	1	6127		
34				&Вернуть...	1	6128		
35				&Журнал версий...	1	7799		

Рис. 26.5. Вывод информации об элементах управления любого уровня вложенности

Подпрограмма `ListAllControls` перебирает коллекцию `CommandBars`. Текущая ячейка в столбце A отслеживается с помощью переменной `Rng`. Имя текущей командной панели отображается в строке состояния. Кроме того, оно выводится в ячейке, на которую указывает переменная `Rng`. После этого подпрограмма перебирает все элементы управления в пределах текущего объекта `CommandBar`. Для этого используется показанная ниже функции `ListControls`.

В главе 14 рассматривалось получение списка командных панелей в редакторе VBE. Для этого достаточно изменить строку

```
For Each cBar in Application.CommandBars
```

на строку

```
For Each cBar in Application.VBE.CommandBars
```

Функция `ListControls` предоставляет дополнительную информацию о каждом элементе управления и каждом вложенном элементе управления. Информация об элементах управления выводится на листе начиная с ячейки, на которую указывает переменная `Rng`. В результате функция `ListControls` возвращает количество строк информации об элементах управления. Переменная `lOffset` используется для вычисления нового значения `Rng`, в соответствии с которым будет выводиться информация об элементах управления на следующей командной панели.

```
Function ListControls(cbCtl As CommandBarControl, Rng As Range) As Long
  ' Указывает на текущую строку относительно Rng
  Dim lOffset As Long
  ' Командная панель, хранящаяся в cbCtl
  Dim ctlSub As CommandBarControl

  On Error Resume Next
  lOffset = 0
  Rng.Offset(lOffset, 1).Value = cbCtl.Caption
  Rng.Offset(lOffset, 2).Value = cbCtl.Type
  ' Попытаться скопировать пиктограмму элемента управления.
  ' Не вставлять изображение, если копирование привело к
  ' ошибке
  cbCtl.CopyFace
  If Err.Number = 0 Then
    ActiveSheet.Paste Rng.Offset(lOffset, 3)
    Rng.Offset(lOffset, 3).Value = cbCtl.FaceId
  End If
  Err.Clear

  ' Проверить тип элемента управления
  Select Case cbCtl.Type
    Case 1, 2, 4, 6, 7, 13, 18
      ' Ничего не делать для этих элементов управления
    Case Else
      ' Если текущий элемент управления содержит другие
      ' элементы управления, вызывать функцию рекурсивно
      For Each ctlSub In cbCtl.Controls
        lOffset = lOffset + _
            ListControls(ctlSub, Rng.Offset(lOffset, 2))
      Next ctlSub
      lOffset = lOffset - 1
  End Select
  ListControls = lOffset + 1
End Function
```

Функция `ListControls` является рекурсивной. Она вызывает сама себя столько раз, сколько существует вложенных уровней элементов управления. Переменная `lOffset` используется для определения текущей строки, в которую записывается информация об элементе управления. Данный код очень напоминает код функции `ListFirstLevelControls`. При этом функция записывает тип элемента управления, а также надпись, пиктограмму и идентификатор пиктограммы. Большинство элементов управления имеют следующие типы:

- 1 — `msoControlButton`;
- 10 — `msoControlPopup`.

Но иногда встречаются элементы управления других типов:

- 2 — `msoControlEdit`;
- 4 — `msoControlComboBox`;
- 6 — `msoControlSplitDropdown`;
- 7 — `msoControlOCXDropdown`;
- 13 — `msoControlSplitButtonPopup`;
- 18 — `msoControlGrid`.

Для отказа от просмотра списка элементов управления в ситуациях, где это не имеет смысла, используется конструкция `Select Case`.

При обнаружении элемента управления с вложенными элементами функция `ListControls` вызывает сама себя, передавая в качестве значения параметра `Rng` текущее значение `Rng` плюс значение переменной `lOffset`. Кроме этого, текущая точка ввода смещается на два столбца вправо. Функция `ListControls` продолжает вызывать сама себя, пока не доберется до самого нижнего уровня вложенности. После этого функция продолжает обрабатывать более верхние уровни элементов управления. При каждом вызове функция возвращает количество строк, записанных относительно `Rng`.

Идентификаторы FaceId

Нижеприведенный код позволяет получить таблицу со списком пиктограмм встроенных элементов управления (рис. 26.6). В Office 97 присутствует около 3500 пиктограмм. В Office 2000 предоставляется 5500 пиктограмм, в Office 2002 — 7000 пиктограмм, а в Office 2003 — 10000 пиктограмм. Обратите внимание, что многие значения `FaceId` соответствуют пустой пиктограмме. Кроме этого, некоторые пиктограммы повторяются с увеличением значения `FaceId`.

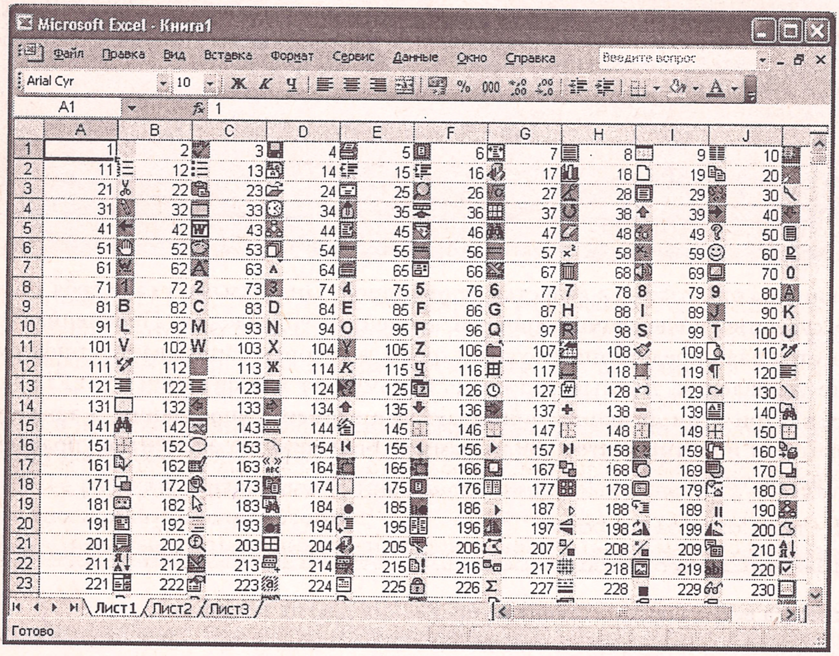


Рис. 26.6. Пиктограммы встроенных элементов управления

Следующий код позволяет вывести список всех пиктограмм встроенных элементов управления:

```
Sub ListAllFaces()
    Dim i As Integer ' Отслеживает текущее значение FaceId
    Dim j As Integer ' Отслеживает текущий столбец листа
    Dim k As Integer ' Отслеживает текущую строку листа
```

```
    Dim cbCtl As CommandBarControl

    Dim cbBar As CommandBar

    If Not IsEmptyWorksheet(ActiveSheet) Then Exit Sub
    On Error GoTo Recover
    Application.ScreenUpdating = False
    ' Создать временную командную панель с единственной
    ' кнопкой для хранения пиктограммы, которая будет скопирована
    ' на лист
    Set cbBar = CommandBars.Add(Position:=msoBarFloating, _
                                MenuBar:=False, _
                                temporary:=True)
    Set cbCtl = cbBar.Controls.Add(Type:=msoControlButton, _
                                   temporary:=True)
    k = 1
    Do
        For j = 1 To 10
            i = i + 1
            Application.StatusBar = "FaceID = " & i

            ' Назначить кнопке текущий FaceId
            cbCtl.FaceId = i
            ' Попытаться скопировать изображение на лист
            cbCtl.CopyFace
            ActiveSheet.Paste Cells(k, j + 1)
            Cells(k, j).Value = i
            ' Разрешить обработку сообщений Windows
            DoEvents
        Next j
        k = k + 1
        ' Разрешить обработку сообщений Windows
    DoEvents
    Loop
    Application.StatusBar = "Всего элементов =" & i

Recover:
    If Err.Number = 1004 Then Resume Next
    Application.StatusBar = False
    cbBar.Delete
End Sub
```

> Обратите внимание, что в этом коде используется две команды DoEvents. При этом Excel возвращает управление и позволяет обрабатывать стандартные сообщения операционной системы Windows. Без этих команд окно Excel выглядело бы зависшим, хотя на самом деле в это время Excel обрабатывает свойства FaceId. В процессе работы этого кода в строке состояния Excel будет выводиться счетчик обработанных элементов управления.

Функция ListAllFaces создает временную панель инструментов cbBar. Для этого применяется метод Add коллекции CommandBars. Созданная панель инструментов объявляется как:

- временная (Temporary), а значит, она будет удалена при завершении сеанса работы в Excel, если не была удалена ранее;
- плавающая (Floating), а не зафиксированная возле края экрана или выполненная в виде всплывающего меню;
- не панель меню (cbBar является панелью инструментов).

На панель инструментов cbBar с помощью метода Add коллекции Controls добавляется временный элемент управления, который присваивается объектной переменной cbCtl.

Цикл Do... Loop продолжает перебор элементов, пока не останется действительных значений FaceId. Цикл Do...Loop увеличивает счетчик на k, которое представляет номера строк на листе. Для каждой строки значение j меняется от 1 до 10. Переменная j описывает столбцы листа. Значение переменной i увеличивается на единицу на каждой итерации цикла For... Next. Значение переменной i соответствует значению свойства FaceId. Свойству FaceId элемента управления cbCtl присваивается значение i. Полученная пиктограмма копируется на лист.

Некоторые элементы управления имеют пустую пиктограмму, а некоторые пиктограммы просто отсутствуют. Пустые пиктограммы копируются без ошибок, а отсутствующие приводят к появлению ошибки 1004. При появлении этой ошибки код переходит к обработке ошибки по метке Recover. Если номер ошибки 1004, код продолжает выполнение после оператора, который привел к появлению сообщения об ошибке. В результате останется пустая ячейка для хранения изображения пиктограммы, а код переберет все значения свойства FaceId, допустимые в данной версии Office. Выбор последнего значения свойства FaceId приводит к появлению ошибки с номером -2147467259. После этого код очищает строку состояния, удаляет временную командную панель и завершает свою работу.

> Полученная в последних трех примерах информация не документирована компанией Microsoft. Эти данные оказываются очень ценными при модификации существующих структур и поведения командных панелей. Кроме того, эта информация может использоваться в качестве справочника по пиктограммам встроенных элементов управления. В надстройке CBList.xla содержится код этой книги, позволяющий просто генерировать эти списки.

Создание новых меню

Если пользователю необходимо предоставить дополнительную функциональность без удаления стандартных команд, в существующую панель меню Worksheet можно добавить новое меню. На рис. 26.7 показано новое меню Custom, расположенное между меню Окно (Window) и Справка (Help).

Рис. 26.7. Добавление нового меню

Новое меню создается с помощью следующего кода.

```
Public Sub AddCustomMenu()
    Dim cbWSMenuBar As CommandBar
    Dim muCustom As CommandBarControl
    Dim iHelpIndex As Integer

    Set cbWSMenuBar = Application.CommandBars("Worksheet Menu Bar")
    iHelpIndex = cbWSMenuBar.Controls("Справка").Index
    Set muCustom = cbWSMenuBar.Controls.Add(Type:=msoControlPopup, _
                                           Before:=iHelpIndex)
    With muCustom
      .Caption = "&Custom"
      With .Controls.Add(Type:=msoControlButton)
         .Caption = "&Show Data Form"
```

```
        .OnAction = "ShowDataForm"
    End With
    With .Controls.Add(Type:=msoControlButton)
        .Caption = "&Print Data List"
        .OnAction = "PrintDataList"
    End With
    With .Controls.Add(Type:=msoControlButton)
        .Caption = "Sort Names &Ascending"
        .BeginGroup = True
        .OnAction = "SortList"
        .Parameter = "Asc"
    End With
    With .Controls.Add(Type:=msoControlButton)
        .Caption = "Sort Names &Descending"
        .OnAction = "SortList"
        .Parameter = "Dsc"
    End With
  End With
End Sub
```

Подпрограмма AddCustomMenu создает объектную переменную cbWSMenuBar, которая ссылается на панель меню Worksheet. Если меню необходимо разместить перед существующими меню, нужно знать их порядковый номер. В данном примере показано определение значения свойства Index элемента управления.

Для добавления нового меню в подпрограмме AddCustomMenu используется метод Add коллекции Controls, принадлежащей панели меню. Свойству Type присваивается значение msoControlPopup, что позволяет связывать с новым элементом управления другие элементы управления. Параметр Before позволяет разместить новое меню перед существующим меню Справка (Help). Если не указать позицию меню, оно будет добавлено в конец панели меню. Свойству Caption нового меню присваивается значение &Custom. Символ & не отображается при выводе меню на экран, но указанный после него символ подчеркивается. Это значит, что для активизации меню с клавиатуры можно воспользоваться комбинацией клавиш <Alt+C>.

После этого с помощью метода Add коллекции Controls нового меню в него добавляются команды. Все команды имеют тип msoControlButton, что позволяет связать макрос с любой командой. Свойству Caption каждой команды присваивается соответствующее значение (учитывая комбинации клавиш, выделяемые с помощью символа &). Свойству OnAction каждой команды присваивается имя запускаемого макроса. У первого пункта меню свойство BeginGroup устанавливается в значение True. Над этим пунктом выводится разделительная полоса, выделяющая элемент управления как начало группы. Большинство команд сортировки имеют одинаковое значение свойства OnAction. Для того чтобы отличать команды, свойству Parameter присваивается соответствующая текстовая строка.

Свойство Parameter применяется для хранения текстового значения, а строка может использоваться по-разному. В данном случае строка "Asc" используется для сортировки по возрастанию, а строка "Dsc" — для сортировки по убыванию. Как будет показано далее, процедура SortList применяет эти строки для определения необходимого порядка сортировки.

Макрос OnAction

Свойству OnAction пункта меню Show Data Form присваивается следующий макрос:

```
Private Sub ShowDataForm()
    fmPersonal.Show
End Sub
```

Эта подпрограмма выводит ту же форму данных, которая демонстрировалась в главе, посвященной диалоговым окнам UserForm. С пунктом меню Print Data List связывается следующий макрос:

```
Private Sub PrintDataList()
    Range("Database").PrintPreview
End Sub
```

Подпрограмма PrintDataList включает предварительный просмотр списка, в котором пользователю предоставляется возможность напечатать список.

С пунктом меню Sort связывается следующий макрос:

```
Private Sub SortList()
    Dim iAscDsc As Integer

    Select Case CommandBars.ActionControl.Parameter
      Case "Asc"
         iAscDsc = xlAscending
      Case "Dsc"
         iAscDsc = xlDescending
   End Select
   Range("Database").Sort Key1:=Range("A2"), Order1:=iAscDsc,
Header:=xlYes
End Sub
```

Для получения ссылки на запустивший подпрограмму элемент управления командной панели используется свойство ActionControl коллекции CommandBars. Это свойство напоминает свойство Application.Caller, применяемое в пользовательских функциях для определения вызвавшего функцию объекта Range.

Обладая информацией о вызвавшем объекте элемента управления, подпрограмма SortList может просмотреть значение свойства Parameter для получения дополнительной информации. Если значение свойства Parameter равно "Asc", подпрограмма SortList выполняет сортировку в порядке возрастания. Если свойство Parameter имеет значение "Dsc", подпрограмма выполняет сортировку в порядке убывания. Кроме этого, элементы управления имеют свойство Tag, используемое точно так же, как свойство Parameter, и содержащее еще одну текстовую строку. Свойство Tag можно применять в качестве альтернативы свойству Parameter. Кроме этого, в нем можно хранить вспомогательные данные.

Передача значений параметров

В предыдущем примере свойство Parameter элемента управления использовалось для передачи информации в подпрограмму, связанную со свойством OnAction. Кроме этого, было показано, что для этих же целей можно применять свойство Tag. Если необходимо передавать более двух фрагментов информации, то лучше воспользоваться подпрограммой, принимающей параметры.

Предположим, необходимо передать три элемента данных, например, название продукта, стоимость и цену. Такая подпрограмма может выглядеть следующим образом:

```
Sub ShowProduct(sName As String, dCost As Double, dPrice As Double)
  MsgBox "Продукт: " & sName & vbCr & _
      "Стоимость: " & Format(dCost, "$0.00") & vbCr & _
      "Цена: " & Format(dPrice, "$0.00")
End Sub
```

Для запуска этой подпрограммы из элемента управления командной панели свойству OnAction элемента управления необходимо присвоить следующее значение:

```
'ShowProduct "Apple", 3, 4'
```

Все выражение заключается в одинарные кавычки. Любой строковый параметр в пределах выражения заключается в двойные кавычки. Для присвоения этого выражения свойству OnAction элемента управления, на который ссылается объектная переменная Ctl, можно воспользоваться следующим кодом:

```
Ctl.OnAction = "'ShowProduct ""Apple"", 3, 4'"
```

Комбинация одинарных и двойных кавычек может оказаться сложной для восприятия. Вся строка заключается в двойные кавычки, поэтому внутренние двойные кавычки необходимо указывать дважды.

Удаление меню

Встроенные и собственные элементы управления можно удалять с помощью метода Delete. Следующая подпрограмма удаляет меню Custom:

```
Public Sub RemoveCustomMenu()
   Dim cbWSMenuBar As CommandBar

   On Error Resume Next
   Set cbWSMenuBar = CommandBars("Worksheet Menu Bar")
   cbWSMenuBar.Controls("Custom").Delete
End Sub
```

Оператор On Error защищает от ситуации, когда меню уже удалено.

> Для возврата встроенной командной панели к первоначальному состоянию и поведению можно применить метод Reset. Если пользователи модифицировали командные панели или используют книги и надстройки, модифицирующие командные панели после установки, применение этого метода нежелательно, так как все модификации будут потеряны.

Следующую процедуру обработки события необходимо добавить в модуль ThisWorkbook. В результате меню Custom будет добавляться при открытии книги и удаляться при закрытии.

```
Private Sub Workbook_BeforeClose(Cancel As Boolean)
    Call RemoveCustomMenu
End Sub

Private Sub Workbook_Open()
    Call AddCustomMenu
End Sub
```

> Важно понять, что модификации командных панелей сохраняются навсегда. Если не удалить созданное в этом примере меню Custom, оно будет доступно на протяжении текущего сеанса работы с Excel и в течение последующих сеансов. Использование этого меню, пока активна другая книга, может привести к непредсказуемым последствиям.

Создание панели инструментов

Простую панель инструментов с кнопками и раскрывающимися списками можно создать вручную. Но существуют и более сложные элементы управления, например `msoControlEdit`, `msoControlDropdown` и `msoControlComboBox`, полноценное управление которыми возможно только с помощью кода VBA. В данном примере создается панель инструментов с тремя элементами управления.

Первый элемент управления имеет тип `msoControlButton` и запрашивает у пользователя ввод списка данных.

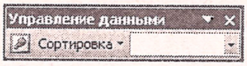


Рис. 26.8. Элемент управления msoControlButton

Второй элемент управления (рис. 26.9) имеет тип `msoControlPopup` и содержит два элемента управления `msoControlButton`.

Третий элемент управления (рис. 26.10) имеет тип `msoControlDropdown` и обеспечивает применение возможности `AutoFilter` к диапазону `Department`.

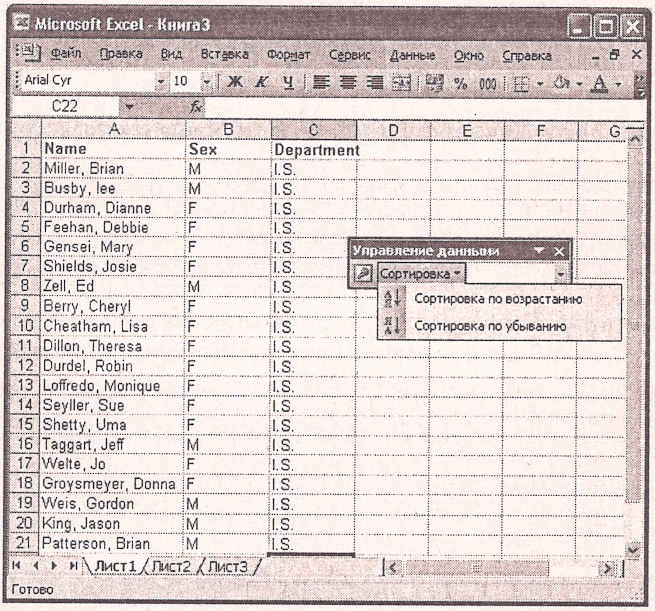


Рис. 26.9. Элемент управления msoControlPopup

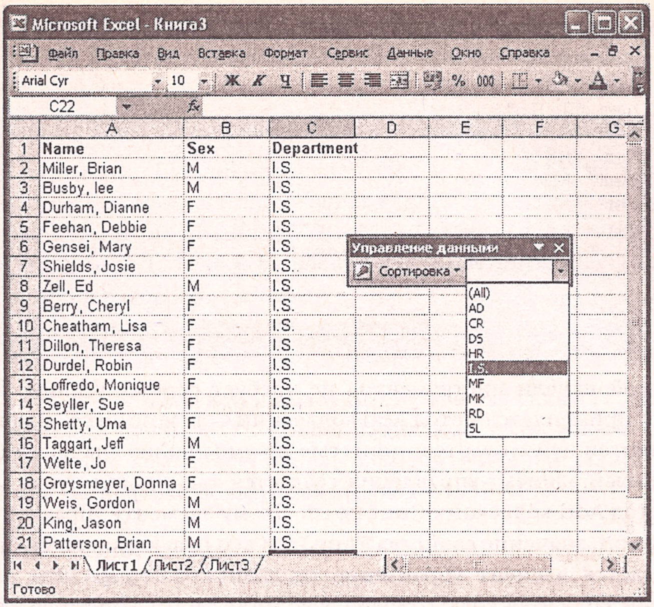


Рис. 26.10. Элемент управления msoControlDropdown

Для создания панели инструментов используется следующий код:

```
Public Sub CreateToolbar()
   ' Удалить существующую панель инструментов,
   ' которая называется Управление данными
On Error Resume Next
   CommandBars("Управление данными").Delete
   On Error GoTo 0
   ' Создать новую панель инструментов
   With CommandBars.Add(Name:="Управление данными")
      With .Controls.Add(Type:=msoControlButton)
         .OnAction = "ShowDataForm"
         .FaceId = 264
         .TooltipText = "Показать форму для данных"
      End With
      With .Controls.Add(Type:=msoControlPopup)
         .Caption = "Сортировка"
         .TooltipText = "Сортировка по убыванию или по возрастанию"
         With .Controls.Add(Type:=msoControlButton)
           .Caption = "Сортировка по возрастанию"
           .FaceId = 210
           .OnAction = "SortList"
           .Parameter = "Asc"
         End With
         With .Controls.Add(Type:=msoControlButton)
           .Caption = "Сортировка по убыванию"
           .FaceId = 211
           .OnAction = "SortList"
           .Parameter = "Dsc"
         End With
      End With
      With .Controls.Add(Type:=msoControlDropdown)
         .AddItem "(All)"
```

```
            .AddItem "AD"
            .AddItem "CR"
            .AddItem "DS"
            .AddItem "HR"
            .AddItem "I.S."
            .AddItem "MF"
            .AddItem "MK"
            .AddItem "RD"
            .AddItem "SL"
            .OnAction = "FilterDepartment"
            .TooltipText = "Выберите отдел"
        End With
        .Visible = True
    End With
End Sub
```

Создание самой панели инструментов не требует особых усилий. В подпрограмме `CreateToolbar` применяется метод `Add` коллекции `CommandBars`. Для всех параметров панели инструментов используются принятые по умолчанию значения (кроме значения поля `Name`). Первый элемент управления создается так же, как пункт меню. Для этого применяется метод `Add` коллекции `Controls`. Этому элементу управления присваивается макрос `OnAction`, значение свойства `FaceId` и всплывающая подсказка.

Вторым создается элемент управления `msoControlPopup`. Это раскрывающийся список, в который заносятся коды отделов. Свойство `OnAction` этого элемента управления указывает на подпрограмму `FilterDepartment`. Кроме этого, элементу правления назначается всплывающая подсказка. После создания элемента управления свойству `Visible` присваивается значение `True`.

Ниже показан код подпрограммы `FilterDepartment`.

```
Sub FilterDepartment()
    Dim stDept As String

    With CommandBars.ActionControl
        stDept = .List(.ListIndex)
    End With
    If stDept = "(All)" Then
        Range("Database").Parent.AutoFilterMode = False
    Else
        Range("Database").AutoFilter Field:=5, Criteria1:=stDept
    End If
End Sub
```

Элемент управления раскрывающегося списка имеет свойство `List`, представляющее собой массив элементов списка, и свойство `ListIndex`, содержащее номер текущего элемента списка. Свойство `ActionControl` объекта `CommandBars` ссылается на активный в данный момент элемент управления и позволяет быстро получить доступ к свойствам `List` и `ListIndex` для определения необходимого кода отдела. Код используется в соответствующей операции `AutoFilter`. Если выбрать пункт (Все), свойство `AutoFilterMode` родительского листа диапазона `Database` устанавливается в значение `False`. При этом будут удалены все раскрывающиеся списки `AutoFilter` и показаны все скрытые строки.

Желательно запускать подпрограмму `CreateToolbar` из процедуры обработки события `Workbook_Open`, а удаление панели инструментов — из процедуры обработки события `Workbook_BeforeClose`. Если панель инструментов не удалить, она постоянно будет присутствовать в Excel. Поведение панели инструментов в пределах другой книги

не определено. Если к коллекции CommandBars обращаться непосредственно из процедур обработки событий, ссылку необходимо квалифицировать с помощью названия объекта Application:

```
Application.CommandBars("Управление данными").Delete
```

Обычно созданная командная панель выводится в том месте, на которое указывает операционная система Windows. Для получения единообразного внешнего вида и указания точного положения новой командной панели можно воспользоваться соответствующими вызовами Win32 API. Для этого в проект VBA придется вставить еще один модуль. Добавьте следующий оператор в новый модуль:

```
Declare Function GetSystemMetrics32 Lib "User32" _
    (ByVal Index As Long) As Long
```

В операторе Declare определяется вызов GetSystemMetrics32, позволяющий получить размеры экрана в пунктах. Эта информация дает возможность определить центральную точку и вывести командную панель. Измените код командной панели следующим образом (в этом варианте используются вызовы Win32 API):

```
Public Sub CreateToolbar()
   ' Удалить существующую панель инструментов,
   ' которая называется Управление данными
   On Error Resume Next
   CommandBars("Управление данными").Delete
   On Error GoTo 0

   ' Создание новой панели инструментов
    With CommandBars.Add(Name:="Управление данными")
       With .Controls.Add(Type:=msoControlButton)
          .OnAction = "ShowDataForm"
          .FaceId = 264
          .TooltipText = "Показать форму для данных"
       End With

       With .Controls.Add(Type:=msoControlPopup)
          .Caption = "Сортировка"
          .TooltipText = "Сортировка по убыванию или по возрастанию"
          With .Controls.Add(Type:=msoControlButton)
             .Caption = "Сортировка по возрастанию"
             .FaceId = 210
             .OnAction = "SortList"
             .Parameter = "Asc"
          End With
          With .Controls.Add(Type:=msoControlButton)
             .Caption = "Сортировка по убыванию"
             .FaceId = 211
             .OnAction = "SortList"
             .Parameter = "Dsc"
          End With
       End With

       With .Controls.Add(Type:=msoControlDropdown)
          .AddItem "(Все)"
          .AddItem "AD"
          .AddItem "CR"
          .AddItem "DS"
          .AddItem "HR"
          .AddItem "ИС"
```

```
            .AddItem "MF"
            .AddItem "MK"
            .AddItem "RD"
            .AddItem "SL"
            .OnAction = "FilterDepartment"
            .TooltipText = "Выберите отдел"
        End With
        .Visible = True
    End With
    Dim lWidth As Long
    Dim lHeight As Long
    lWidth = Module1.GetSystemMetrics32(0) ' Ширина экрана в пунктах
    lHeight = Module1.GetSystemMetrics32(1) ' Высота экрана в пунктах
    With CommandBars("Управление данными")
      .Position = msoBarFloating
      .Left = lWidth / 2 - .Width / 2
      .Top = lHeight / 2 - .Height / 2
    End With
End Sub
```

В блоке кода определяются две переменные, используемые для хранения высоты и ширины экрана. Для получения информации о высоте и ширине экрана используется вызов Win32 API GetSystemMetrics32. Параметры функции позволяют получить разную информацию. Обладая информацией о размере экрана, можно произвести соответствующие вычисления и определить свойства командной панели для отображения на экране. Использование API из кода VBA позволяет значительно расширить возможности интерпретатора.

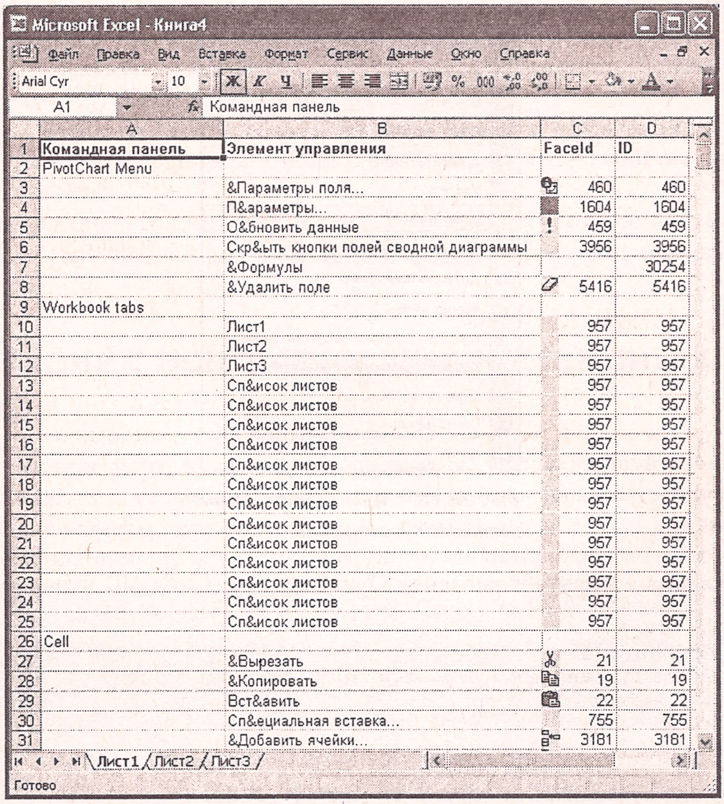
Контекстные меню

Встроенные контекстные меню Excel выводятся в результате работы подпрограммы ListFirstLevelControls, которая рассматривалась ранее в этой главе. Модифицированная версия этой подпрограммы отображает список командных панелей типа msoBarTypePopup (рис. 26.11).

Ниже показан код для вывода списка контекстных меню.

```
Sub ListPopups()

    Dim cbCtl As CommandBarControl
    Dim cbBar As CommandBar
    Dim i As Integer

    If Not IsEmptyWorksheet(ActiveSheet) Then Exit Sub
    On Error Resume Next
    Application.ScreenUpdating = False
    Cells(1, 1).Value = "Командная панель"
    Cells(1, 2).Value = "Элемент управления"
    Cells(1, 3).Value = "FaceId"
    Cells(1, 4).Value = "ID"
    Cells(1, 1).Resize(1, 4).Font.Bold = True
    i = 2
  For Each cbBar In CommandBars
      Application.StatusBar = "Обработка панели " & cbBar.Name
      If cbBar.Type = msoBarTypePopup Then
         Cells(i, 1).Value = cbBar.Name
```

```
            i = i + 1
            For Each cbCtl In cbBar.Controls
               Cells(i, 2).Value = cbCtl.Caption
               cbCtl.CopyFace
               If Err.Number = 0 Then
                  ActiveSheet.Paste Cells(i, 3)
                  Cells(i, 3).Value = cbCtl.FaceID
               End If
               Cells(i, 4).Value = cbCtl.ID
               Err.Clear
               i = i + 1
            Next cbCtl
         End If
   Next cbBar
   Range("A:B").EntireColumn.AutoFit
   Application.StatusBar = False
End Sub
```



	A	B	C	D
1	Командная панель	Элемент управления	FaceId	ID
2	PivotChart Menu			
3		&Параметры поля...	460	460
4		П&араметры...	1604	1604
5		О&бновить данные	459	459
6		Скр&ыть кнопки полей сводной диаграммы	3956	3956
7		&Формулы		30254
8		&Удалить поле	5416	5416
9	Workbook tabs			
10		Лист1	957	957
11		Лист2	957	957
12		Лист3	957	957
13		Сп&исок листов	957	957
14		Сп&исок листов	957	957
15		Сп&исок листов	957	957
16		Сп&исок листов	957	957
17		Сп&исок листов	957	957
18		Сп&исок листов	957	957
19		Сп&исок листов	957	957
20		Сп&исок листов	957	957
21		Сп&исок листов	957	957
22		Сп&исок листов	957	957
23		Сп&исок листов	957	957
24		Сп&исок листов	957	957
25		Сп&исок листов	957	957
26	Cell			
27		&Вырезать	21	21
28		&Копировать	19	19
29		Вст&авить	22	22
30		Сп&ециальная вставка...	755	755
31		&Добавить ячейки...	3181	3181

Рис. 26.11. Список встроенных контекстных меню Excel

Этот код практически ничем не отличается от кода подпрограммы `ListFirstLevelControls`. Отличается только первая структура `If`, которая заставляет подпрограмму обрабатывать только элементы управления `msoBarTypePopup`. Если рассмотреть список, полученный в результате работы подпрограммы `ListPopups`, можно

узнать распространенные контекстные меню. Например, в списке показаны контекстные меню `Cell`, `Row` и `Columns`, соответствующие контекстным меню, появляющимся при щелчке правой кнопкой мыши на ячейке листа, номере строки и букве столбца.

Присутствие двух экземпляров контекстных меню Cell, Row и Column может показаться странным. Первый набор используется, когда лист находится в нормальном режиме, а второй — в режиме Page Break Preview.

> Еще одной интересной панелью является командная панель Workbook Tabs. Это не то контекстное меню, которое возникает при щелчке правой кнопкой мыши на вкладках листов. Это контекстное меню используется вместе с кнопками перемещения слева от вкладок листов. Для вкладок применяется контекстное меню Ply.

После идентификации контекстного меню с помощью VBA его можно модифицировать для собственных задач. Например, на рис. 26.12 показано модифицированное контекстное меню Cell, в котором присутствует команда Очистить все (Clear All).

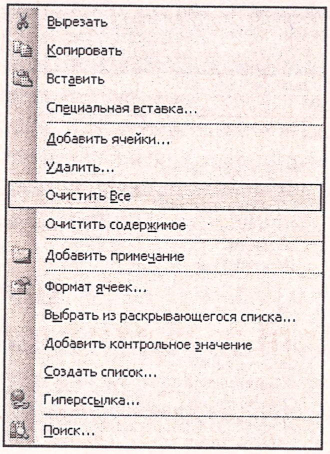


Рис. 26.12. Модифицированное контекстное меню Cell

Для добавления элемента управления Очистить все (Clear All) используется следующий код:

```
Public Sub AddShortCut()
   Dim cbBar As CommandBar
   Dim cbCtl As CommandBarControl
   Dim lIndex As Long

   Set cbBar = CommandBars("Cell")
   lIndex = cbBar.Controls("Clear Contents").Index
   Set cbCtl = cbBar.Controls.Add(Type:=msoControlButton, _
                              ID:=1964, Before:=lIndex)
   cbCtl.Caption = "Очистить &Все"
End Sub
```

В начале выполнения подпрограммы `AddShortCut` переменной `cbBar` присваивается ссылка на командную панель `Cell`.

Если необходима ссылка на командную панель Cell, которая отображается в представлении Page Break в Excel 2003, можно воспользоваться свойством Index:

```
Set cbBar = CommandBars(31)
```

Если код должен быть совместим с другими версиями Office, то в данном случае необходимо соблюдать осторожность. В Excel 2000 свойство Index командной панели Cell из представления Page Break имеет значение 26, а в Excel 97 это же свойство имеет значение 24.

Подпрограмма AddShortCut записывает свойство Index элемента управления Очистить содержимое (Clear Contents) в переменную lIndex, что позволяет добавить новый элемент управления перед элементом управления Очистить содержимое (Clear Contents). Для добавления нового элемента управления в командную панель cbBar в подпрограмме AddShortCut используется метод Add коллекции Controls. В качестве параметра метода передается значение свойства Id встроенного пункта меню Правка⇨Очистить⇨Все (Edit⇨Clear⇨ All).

Метод Add коллекции Controls позволяет указывать свойство Id встроенных команд. Значение свойства Id можно узнать с помощью подпрограммы ListAllControls. Значение свойства Id совпадает со значением свойства FaceId. Для пункта Правка⇨Очистить⇨Все (Edit⇨Clear⇨ All) свойство Id равно 1964.

Встроенное свойство Caption добавленного элемента управления имеет значение "Все" ("All"), поэтому в подпрограмме AddShortCut значение свойства Caption меняется на более понятное.

Модифицированное контекстное меню Cell можно спокойно оставлять в коллекции CommandBars. Меню не связано с книгой и не зависит от доступа к макросам в пределах определенной книги.

Отображение всплывающих командных панелей

Если контекстное меню должно отображаться без щелчка правой кнопкой мыши на ячейке или диаграмме, то для его отображения можно воспользоваться несколькими вариантами кода. Например, для отображения контекстного меню Cell можно воспользоваться комбинацией клавиш <Ctrl+Shift+C>. Для этого применяется следующий код:

```
Sub SetShortCut()
   Application.OnKey "^+c", "ShowCellShortCut"
End Sub
Private Sub ShowCellShortCut()
   CommandBars("Cell").ShowPopup x:=0, y:=0
End Sub
```

Для отображения контекстного меню Cell в левом верхнем углу экрана в подпрограмме ShowCellShortCut используется метод ShowPopup. Значения координат экрана x и y передаются методу в качестве параметров.

Кроме этого, всплывающее меню можно полностью создать самостоятельно. Всплывающее меню f (рис. 26.13) отображается при щелчке правой кнопкой мыши внутри диапазона Database. За пределами этого диапазона выводится обычное контекстное меню Cell.

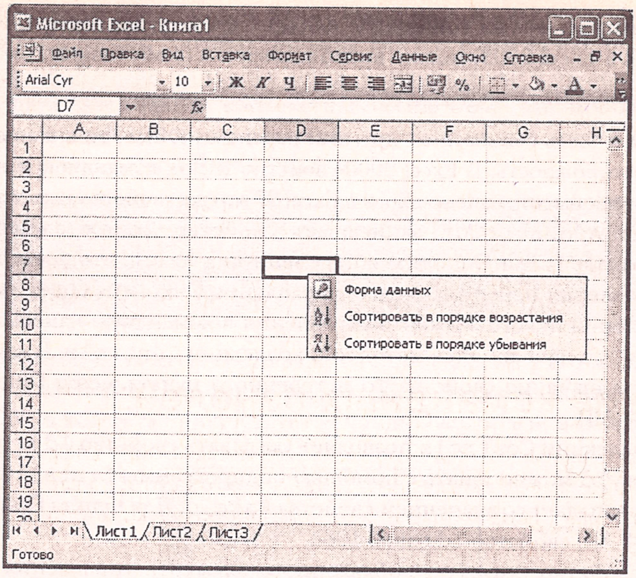


Рис. 26.13. Отображение собственного контекстного меню

Для создания контекстного меню используется следующий код:

```
Sub MakePopup()
   With CommandBars.Add(Name:="Data Popup", Position:=msoBarPopup)

      With .Controls.Add(Type:=msoControlButton)
         .OnAction = "ShowDataForm"
         .FaceID = 264
         .Caption = "Форма данных"
         .TooltipText = "Показать форму данных"
      End With
      With .Controls.Add(Type:=msoControlButton)
         .Caption = "Сортировать в порядке возрастания"
         .FaceID = 210
         .OnAction = "SortList"
         .Parameter = "Asc"
      End With
      With .Controls.Add(Type:=msoControlButton)
         .Caption = "Сортировать в порядке убывания"
         .FaceID = 211
         .OnAction = "SortList"
         .Parameter = "Dsc"
      End With
   End With
End Sub
```

Этот код напоминает код для создания меню и панели инструментов из предыдущих примеров. Разница заключается в том, что при создании контекстного меню с помощью метода Add коллекции CommandBars параметр Position устанавливается в значение msoBarPopup. В данном случае свойство Name устанавливается в значение Data Popup.

Для отображения контекстного меню можно воспользоваться следующей процедурой обработки события BeforeDoubleClick, расположенной в модуле кода листа, в котором находится диапазон Database.

```
Private Sub Worksheet_BeforeRightClick(ByVal Target As Range, _
                                       Cancel As Boolean)
   If Not Intersect(Range("Database"), Target) Is Nothing Then
      CommandBars("Data Popup").ShowPopup
      Cancel = True
   End If
End Sub
```

При щелчке правой кнопкой мыши на листе процедура обработки события проверяет попадание ячейки Target в диапазон Database. Если ячейка находится в пределах диапазона, подпрограмма выводит контекстное меню Data Popup и отменяет событие щелчка правой кнопкой мыши. В противном случае отображается стандартное контекстное меню Cell.

Отключение командных панелей

Командные панели имеют свойство Enabled и свойство Visible. Если командная панель включена и не имеет тип msoBarTypePopup, она отображается в диалоговом окне Сервис⇨Настройка (Tools⇨Customize). Если напротив командной панели в диалоговом окне Сервис⇨Настройка (Tools⇨Customize) установлен флажок, панель отображается на экране.

Если свойство Enabled не установлено в значение True, установить свойство Visible в значение True невозможно. Установка свойства Visible командной панели типа msoBarTypeNormal в значение False удалит панель с экрана. Установка свойства Visible панели меню Worksheet в значение False не работает. Excel рассматривает этот объект как особый случай и отображает его всегда вместе с активным листом. Единственным способом удалить с экрана меню Worksheet является установка свойства Enabled в значение False.

Следующий код удаляет с экрана все видимые панели инструментов и панель меню Worksheet:

```
Sub RemoveToolbarsAndWorksheetMenuBar()
   Dim cbBar As CommandBar
   For Each cbBar In CommandBars
      If cbBar.Enabled And cbBar.Type = msoBarTypeNormal Then
         cbBar.Visible = False
      End If
   Next cbBar
   CommandBars("Worksheet Menu Bar").Enabled = False
   Application.OnKey "%-", ""
End Sub
```

Результат работы кода показан ниже (рис. 26.14).

> Последняя операция в подпрограмме RemoveToolbarsAndWorksheetMenuBar отключает комбинацию клавиш <Alt+->, нажатие которой приводит к отображению управляющего меню окна книги. Если не отключить эту комбинацию клавиш, пользователь сможет обращаться к управляющему меню и использовать клавиши управления курсором для медленного появления фантомной копии панели меню Worksheet.

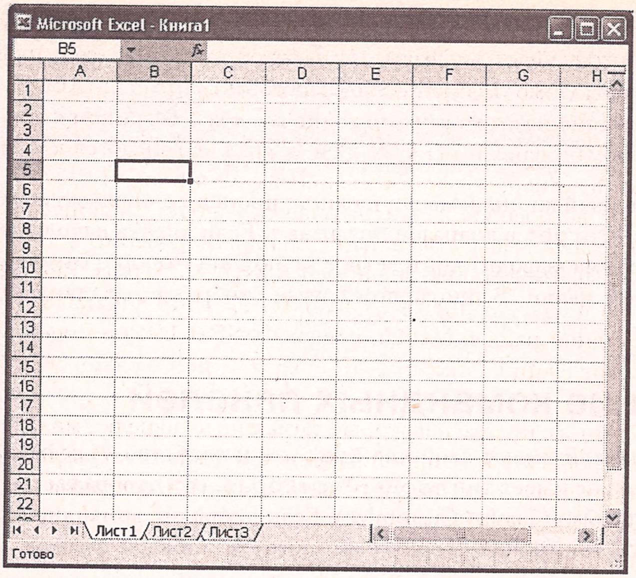


Рис. 26.14. Удаление с экрана всех панелей инструментов и панели меню Worksheet

Для восстановления панели меню Worksheet и панелей инструментов Стандартная (Standard) и Форматирование (Formatting) можно воспользоваться следующим кодом. Предполагается, что свойство `Enabled` панелей инструментов не установлено в значение `False`:

```
Sub RestoreToolbarsAndWorksheetMenuBar()
    CommandBars("Worksheet Menu Bar").Enabled = True
    Application.OnKey "%-"
    CommandBars("Standard").Visible = True
    CommandBars("Formatting").Visible = True
End Sub
```

Отключение комбинации клавиш для доступа к диалоговому окну Настройка

Если пользователям необходимо запретить доступ к собственным или встроенным командным панелям, можно ограничить доступ к диалоговому окну и панели инструментов Настройка (Customize). Следующий код должен быть сохранен в личной книге макросов (`Personal.xls`). После этого он будет запускаться при каждом запуске Excel.

```
Private Sub Workbook_Open()
    ' Здесь находится код модификации командных панелей
    Application.CommandBars("Tools").Controls("Customize...").
↳Enabled = False
    Application.CommandBars("Toolbar List").Enabled = False
End Sub
```

В первой строке кода отключается пункт меню Сервис⇨Настройка (Tools⇨Customize). Во второй строке отключается контекстное меню, которое отображается при щелчке пра-

вой кнопкой мыши на командной панели. Кроме этого, отключается пункт меню Вид⇨Панели инструментов (View⇨Toolbars). Так как этот код находится в процедуре обработки события книги, необходимо указывать ссылку на объект Application.

Обратите внимание на синтаксис показанного кода. В данном случае элемент управления Сервис (Tools) в панели меню Worksheet рассматривается как отдельная командная панель. Поиск в сгенерированном подпрограммой ListAllControls списке покажет, что существует встроенная командная панель Menus. К элементам этой командной панели можно обращаться как к отдельным командным панелям.

Командная панель Toolbar List появилась в Excel 97 Service Release. Эта командная панель не доступна в более ранней версии Excel 97 — это специальная скрытая панель. Как элемент командной панели Menus, панель ToolBar List не имеет соответствующего значения Index коллекции CommandBars, хотя к ней можно обращаться по имени (параметр Name).

Если необходимо защитить некоторые командные панели, можно воспользоваться свойством Protect объекта командной панели. Следующий код включает все параметры защиты панели инструментов Стандартная (Standard). Константы для ненужных параметров защиты можно не указывать:

```
Sub ProtectToolbar()
   CommandBars("Standard").Protection = msoBarNoCustomize + _
                                        msoBarNoResize + _
                                        msoBarNoMove + _
                                        msoBarNoChangeVisible + _
                                        msoBarNoChangeDock + _
                                        msoBarNoVerticalDock + _
                                        msoBarNoHorizonaldock
End Sub
```

Для отключения защиты воспользуйтесь следующим кодом:

```
Sub UnProtectToolbar()
   CommandBars("Standard").Protection = msoBarNoProtection
End Sub
```

Создание командных панелей на основе таблиц

Профессиональные разработчики Excel редко создают панели инструментов и пункты меню по одному. Чаще всего используется решение на основе таблиц, в результате которого в таблицу заносится информация о необходимых элементах и специальная подпрограмма генерирует все элементы на основе этой информации. Такой подход значительно упрощает определение и проектирование командных панелей.

Предположим, что созданное ранее в этой главе меню Custom необходимо создать с помощью нового метода. Сначала создайте таблицу с информацией о меню. Вставьте новый лист, переименуйте лист в MenuTable и заполните его, как показано на рис. 26.15 и рис. 26.16. На листе Data содержится база данных работников. Список данных будет использоваться для определения списка отделов.

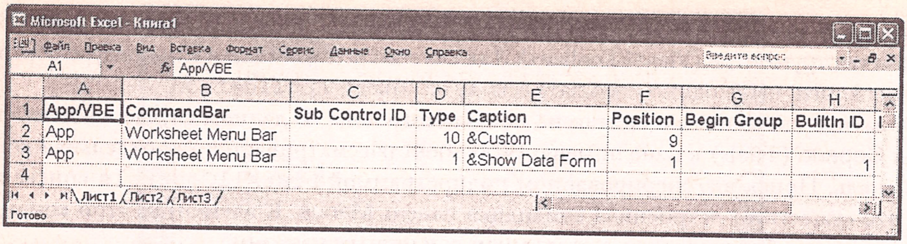


Рис. 26.15. Создание меню на основе таблиц

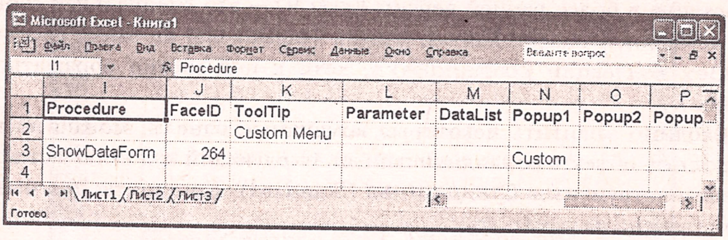


Рис. 26.16. Создание меню на основе таблиц

На листе MenuTable используются следующие столбцы:

Столбец	*Название*	*Описание*
A	App/VBE	Значение App заставляет подпрограмму добавлять пункт в меню Excel. Значение VBE заставляет подпрограмму добавлять пункт в меню редактора VBE. Код обработки пунктов меню VBE рассматривался в главе 23
B	CommandBar	Имя командой панели верхнего уровня, в которую добавляется создаваемое меню. Эти имена извлекаются из листинга, который генерировался ранее в этой главе
C	Sub Control ID	Номер ID встроенного всплывающего меню, с которым связывается добавляемое меню. Например, 30002 является идентификатором всплывающего меню Файл (File)
D	Type	Тип добавляемого элемента управления: 1 — нормальная кнопка, 10 — контекстное меню и т.д. Эти значения соответствуют значениям констант `msoControl...`, показанным в окне Object Browser (Просмотр объектов)
E	Caption	Текст пункта меню
F	Position	Положение добавляемого пункта меню на командной панели. Оставьте это поле пустым для добавления меню в конец панели
G	Begin Group	Значения `True` или `False` определяют добавление разделителя перед пунктом меню
H	BuiltIn ID	Если добавляется встроенный пункт меню, в этом поле указывается идентификатор меню. Для собственных пунктов меню используется идентификатор 1

Столбец	*Название*	*Описание*
I	Procedure	Имя подпрограммы, которая запускается при щелчке на пункте меню
J	FaceId	Идентификатор встроенной пиктограммы элемента управления, которая будет связана с добавляемым пунктом меню. Кроме этого, в данном поле может указываться имя изображения на листе, которое будет использоваться в качестве пиктограммы кнопки. Стандартная пиктограмма Создать (New) имеет идентификатор 18
K	ToolTip	Текст всплывающей подсказки для кнопки
L	Parameter	Строка, которая присваивается свойству `Parameter` создаваемого элемента управления
M	DataList	Используется только для элементов управления, содержащих раскрывающиеся списки, например `msoControlDropdown` (тип 3). В поле содержится имя диапазона ячеек листа, которые называются `DataList` и содержат элементы создаваемого раскрывающегося списка
N	Popup1-n	При добавлении собственных контекстных меню значение этого поля используется в качестве заголовка контекстного меню. Можно добавлять любое количество уровней вложенности. Для этого необходимо добавить нужное количество столбцов. Код автоматически определит дополнительные столбцы

Так как лист MenuTable будет часто использоваться в коде, желательно присвоить ему осмысленное имя, например `shMenuTable`. Для этого найдите и выделите лист в окне Project Explorer (Окно проекта) в редакторе VBE и измените имя листа в окне Properties (Свойства). После этого в окне Project Explorer (Окно проекта) имя объекта будет выглядеть как shMenuTable (MenuTable). Использование такого имени позволит обращаться к листу непосредственно, как к объекту. В результате следующие две строки будут эквивалентны:

```
Debug.Print ThisWorkbook.Worksheets("MenuTable").Name
Debug.Print shMenuTable.Name
```

Лист `DataLists` должен быть переименован в `shDataLists`.

Ниже показан код для создания меню на основе информации в таблице. Код должен находиться в новом модуле, который называется `modSetupBars`.

В начале модуля определяется большое количество констант. Они соответствуют каждому столбцу таблицы меню и используются на протяжении всего кода. Если структура меню изменится, достаточно переименовать эти константы и не нужно будет искать "магические числа" по всему коду.

```
' Константы для столбцов таблицы с описанием командных панелей
Const miTABLE_APP_VBE              As Integer = 1
Const miTABLE_COMMANDBAR_NAME      As Integer = 2
Const miTABLE_CONTROL_ID           As Integer = 3
Const miTABLE_CONTROL_TYPE         As Integer = 4
Const miTABLE_CONTROL_CAPTION      As Integer = 5
Const miTABLE_CONTROL_POSITION     As Integer = 6
Const miTABLE_CONTROL_GROUP        As Integer = 7
```

```
Const miTABLE_CONTROL_BUILTIN   As Integer = 8
Const miTABLE_CONTROL_PROC      As Integer = 9
Const miTABLE_CONTROL_FACEID    As Integer = 10
Const miTABLE_CONTROL_TOOLTIP   As Integer = 11
Const miTABLE_CONTROL_PARAMETER As Integer = 12
Const miTABLE_CONTROL_DATALIST  As Integer = 13
Const miTABLE_POPUP_START       As Integer = 14

' Константа для определения времени жизни командной панели
' Если установить эту константу в значение False, пользователи
' не потеряют дополнительные элементы управления, добавленные
' на вашу командную панель
Const mbTEMPORARY               As Boolean = False

' Следующий идентификатор приложения используется для идентификации
' меню, что значительно упрощает их удаление
Const psAppID As String = "TableDrivenCommandBars"
```

Константа mbTEMPORARY позволяет сделать изменения меню постоянными или временными. Константа psAppID содержит строку идентификации, которую можно присваивать свойству Tag создаваемых элементов управления. Такая идентификация значительно упрощает поиск и удаление элементов управления.

Подпрограмма фактического создания меню вызывается из процедуры Auto_Open для книги или процедуры обработки события Workbook_Open.

```
' Подпрограмма: SetUpMenus
' Назначение:    Добавляет командные панели, определенные на
' листе shMenuTable

Sub SetUpMenus()
   Dim rgRow As Range
   Dim cbAllBars As CommandBars
   Dim cbBar As CommandBar
   Dim cbBtn As CommandBarControl
   Dim iBuiltInID As Integer, iPopUpCol As Integer, vaData As Variant

   On Error Resume Next  'Игнорировать ошибки в таблице определений

   ' Удалить все меню перед  созданием.
   ' Такое решение защищает от дублирования меню.
   RemoveMenus

   ' Перебрать все строки таблицы генерации меню
   For Each rgRow In shMenuTable.Cells(1).CurrentRegion.Rows
      ' Игнорировать строку с заголовками
      If rgRow.Row > 1 Then
         ' Считать строку в массив со значениями ячеек
         vaData = rgRow.Value

         Set cbBar = Nothing
```

Для добавления меню в Excel и в редактор VBE может использоваться единая подпрограмма. Единственное различие заключается в применяемой коллекции CommandBars (может использоваться коллекция VBE или Excel). В данном примере кода не содержатся все необходимые элементы для создания меню в VBE. Дополнительные требования для создания таких меню рассматривались в главе 23.

```
' Получить коллекцию всех командных панелей в VBE или Excel
    If vaData(1, miTABLE_APP_VBE) = "VBE" Then
       Set cbAllBars = Application.VBE.CommandBars
    Else
       Set cbAllBars = Application.CommandBars
    End If

    ' Попытаться найти интересующую командную панель
    Set cbBar = cbAllBars.Item(vaData(1, miTABLE_COMMANDBAR_NAME))

    ' Если панель не найдена, ее необходимо создать!
    If cbBar Is Nothing Then
       Set cbBar = cbAllBars.Add(_
           Name:=vaData(1, miTABLE_COMMANDBAR_NAME), _
                                          temporary:=mbTEMPORARY)
    End If
```

Если необходимо найти встроенное всплывающее меню, в которое будет добавляться элемент управления, можно выполнить рекурсивный поиск в коллекции CommandBars. Например, если нужно добавить пункт меню в меню Правка⇨Очистить (Edit⇨Clear), можно ввести идентификатор меню Очистить (Clear) (30021) в столбец Sub Control ID. С другой стороны, в столбцы PopUp можно ввести одно или несколько имен элементов управления. Ввод **Edit** в разделе PopUp1 и **Clear** в PopUp2 позволяет добиться того же результата, что и ввод 30021 в поле Sub Control ID. Первый метод удобнее при добавлении элементов управления в существующие меню. Второй метод используется при добавлении элементов управления в создаваемые меню.

```
    ' Если значение установлено, найти встроенное всплывающее
    ' меню (по идентификатору), в которое будет добавляться
    ' элемент управления. Например,  Worksheet Menu Bar⇨Edit
    If Not IsEmpty(vaData(1, miTABLE_CONTROL_ID)) Then
       Set cbBar = cbBar.FindControl(ID:=vaData(1, _
                  miTABLE_CONTROL_ID), Recursive:=True).CommandBar
    End If
    ' Перебрать все имена в столбце PopUp для перехода
    ' по структуре меню
    For iPopUpCol = miTABLE_POPUP_START To UBound(vaData, 2)
       ' Если значение установлено, перейти к следующему
       ' всплывающему меню.
       If Not IsEmpty(vaData(1, iPopUpCol)) Then
          Set cbBar = cbBar.Controls(vaData(1,
↳iPopUpCol)).CommandBar
       End If
    Next
```

При добавлении существующего элемента управления Excel значение свойства Id можно указать в столбце BuiltIn ID. Если элемент управления должен вызывать собственную процедуру, имя процедуры указывается в поле Procedure:

```
    ' Получить идентификатор добавляемого встроенного элемента
    ' управления
    iBuiltInID = vaData(1, miTABLE_CONTROL_BUILTIN)

    ' Если идентификатор не указан, сделать его равным 1, что
    ' соответствует создаваемому элементу управления
    If iBuiltInID = 0 Then iBuiltInID = 1

    ' Добавить элемент управления в командную панель
```

```
    If IsEmpty(vaData(1, miTABLE_CONTROL_POSITION)) Or _
                vaData(1, miTABLE_CONTROL_POSITION) > _
                cbBar.Controls.Count Then
        Set cbBtn = cbBar.Controls.Add(Type:=vaData(1, _
                    miTABLE_CONTROL_TYPE), ID:=iBuiltInID, _
                    temporary:= mbTEMPORARY)
    Else
        Set cbBtn = cbBar.Controls.Add(Type:=vaData(1, _
                    miTABLE_CONTROL_TYPE), ID:=iBuiltInID, _
                    temporary:= mbTEMPORARY, _
                    before:=vaData(1, _
                    miTABLE_CONTROL_POSITION))
    End If

    ' Установить остальные свойства кнопки
    With cbBtn
        .Caption = vaData(1, miTABLE_CONTROL_CAPTION)
        .BeginGroup = vaData(1, miTABLE_CONTROL_GROUP)
        .TooltipText = vaData(1, miTABLE_CONTROL_TOOLTIP)
```

Можно использовать одну из стандартных пиктограмм Office (для этого необходимо указать значение FaceId) или предоставить собственное изображение, применяемое в качестве пиктограммы. Для использования собственного изображения в столбце FaceId достаточно указать имя объекта Picture.

```
        ' Поле FaceId может быть пустым, если кнопка не содержит
        ' пиктограмы, может содержать номер встроенной пиктограммы
        ' или может содержать имя объекта на листе, в котором
        ' хранится изображение.
        If Not IsEmpty(vaData(1, miTABLE_CONTROL_FACEID)) Then
            If IsNumeric(vaData(1, miTABLE_CONTROL_FACEID)) Then
                ' Числовое значение FaceID. Используем его.
                .FaceId = vaData(1, miTABLE_CONTROL_FACEID)
            Else
                ' Текстовое значение FaceID, поэтому копируем
                ' изображение на кнопку.
                shMenuTable.Shapes(vaData(1, _
                    miTABLE_CONTROL_FACEID)).CopyPicture
                .PasteFace
            End If
        End If
```

Желательно для всех своих пунктов меню устанавливать свойство, идентифицирующее тот пункт, который создан разработчиком. Если для этого используется свойство Tag, для поиска всех своих пунктов меню можно применить метод FindControl объекта CommandBars. При этом не нужно помнить, что и куда было добавлено. Этот метод поиска используется в процедуре RemoveMenus далее в этом модуле.

```
        ' Установить свойство Tag для идентификации созданного
        ' элемента управления.
        ' Таким образом созданный элемент управления можно будет
        ' найти даже в случае переименования или перемещения
        ' пользователем.
        .Tag = psAppID

        ' Установить свойство OnAction элемента управления.
        ' Заключить имя книги в кавычки, на случай, если имя
        ' содержит пробелы.
        If Not IsEmpty(vaData(1, miTABLE_CONTROL_PROC)) Then
```

```
        .OnAction = "'" & ThisWorkbook.Name & "'!" & vaData(1, _
                          miTABLE_CONTROL_PROC)
      End If
```

Если процедура ищет информацию в свойстве Parameter элемента управления, эта информация вводится в столбец Parameter:

```
      ' Присвоить значение свойству Parameter
      If Not IsEmpty(vaData(1, miTABLE_CONTROL_PARAMETER)) Then
        .Parameter = vaData(1, miTABLE_CONTROL_PARAMETER)
      End If
```

Для раскрывающегося списка список значений вводится на листе DataLists и списку присваивается имя. Имя можно ввести в столбце DataList:

```
            ' Внести список данных в раскрывающийся список
            If Not IsEmpty(vaData(1, miTABLE_CONTROL_DATALIST)) Then
              For Each Rng In shDatalists.Range(vaData(1, _
                          miTABLE_CONTROL_DATALIST))
                .AddItem Rng.Value
              Next Rng
              End If
          End With
      End If
   Next rgRow
End Sub
```

После закрытия книги приложения необходимо запустить код, который удалит все добавленные меню. Некоторые разработчики используют вызов CommandBars.Reset, но при этом в командных панелях будут удалены все модификации. Более подходящим решением является поиск и удаление всех собственных пунктов меню и командных панелей. Для этого требуется две подпрограммы. Первая удаляет все пункты меню из конкретной коллекции CommandBars (поиск выполняется по значению свойства Tag):

```
Private Sub RemoveMenusFromBars(cbBars As CommandBars)
   Dim cbCtl As CommandBarControl

   ' Игнорировать ошибки при удалении собственных пунктов меню
   On Error Resume Next

   ' Использовать коллекцию CommandBars приложения или редактора VBE
   With cbBars

      ' Найти объект CommandBarControl с нашим значением свойства Tag
      Set cbCtl = .FindControl(Tag:=psAppID)

      ' Выполнять цикл, пока элемент управления не будет найден
      Do Until cbCtl Is Nothing
         ' Удалить найденный элемент управления
         cbCtl.Delete

         ' Найти следующий элемент управления
         Set cbCtl = .FindControl(Tag:=psAppID)
      Loop
   End With
End Sub
```

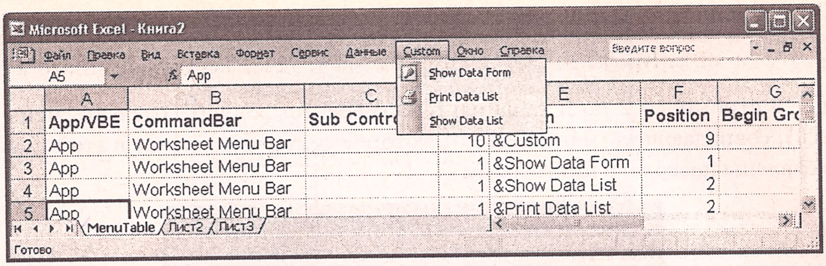
Вторая подпрограмма вызывает первую для удаления пунктов меню из командных панелей Excel и VBE. Кроме этого, подпрограмма удаляет все добавленные командные панели, в которые пользователь не вставил собственные элементы управления:

```
Sub RemoveMenus()
   Dim cbBar As CommandBar, rgRow As Range, stBarName As String

   ' Игнорировать ошибки при удалении пунктов меню
   ' и командных панелей
   On Error Resume Next

   ' Удалять пункты пункты меню из созданных командных
   ' панелей Excel и VBE
   RemoveMenusFromBars Application.CommandBars
   RemoveMenusFromBars Application.VBE.CommandBars

   ' Перебрать все строки таблицы генерации меню
   For Each rgRow In shMenuTable.Cells(1).CurrentRegion.Rows
      ' Игнорировать строку с заголовками
      If rgRow.Row > 1 Then
         stBarName = rgRow.Cells(1, miTABLE_COMMANDBAR_NAME)

         Set cbBar = Nothing
         ' Найти командную панель в VBE или Excel
         If rgRow.Cells(1, miTABLE_APP_VBE) = "VBE" Then
            Set cbBar = Application.VBE.CommandBars(stBarName)
         Else
            Set cbBar = Application.CommandBars(stBarName)
         End If
         ' Если командная панель найдена и не является встроенной,
         ' удалить командную панель
         If Not cbBar Is Nothing Then
            If Not cbBar.BuiltIn Then
               ' Удалять только пустые командные панели -
               ' пользователь или другое приложение могли
               ' добавить на командную панель собственные
               ' элементы управления
               If cbBar.Controls.Count = 0 Then cbBar.Delete
            End If
         End If
      End If
   Next
End Sub
```

Подпрограмму SetUpMenus необходимо запускать из подпрограммы Auto_Open или процедуры обработки события Workbook_Open, а подпрограмму RemoveMenus — из подпрограммы Auto_Close или процедуры обработки события Workbook_BeforeClose.

На этом этапе создан полный шаблон, который может использоваться в качестве основы для любого приложения Excel (или применяться в обычной книге, в которой необходимо модифицировать структуру меню).

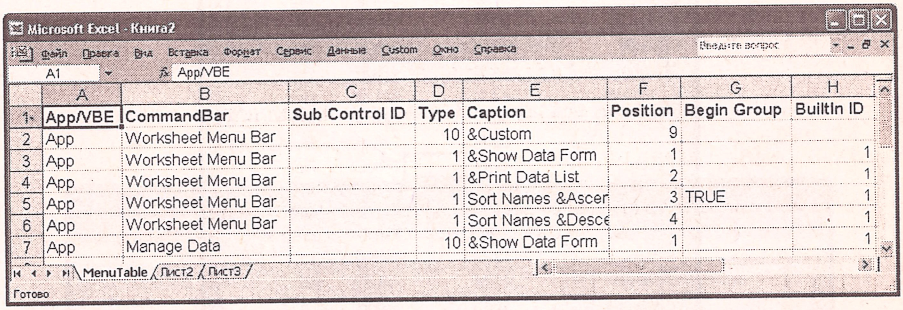
В первой записи таблицы (рис. 26.15 и рис. 26.16) в панель меню Workbook добавляется меню Custom. Вторая запись определяет пункт меню, который называется Show Data Form (рис. 26.17).



	A	B	C	D	E	F	G
1	App/VBE	CommandBar	Sub Contr		n	Position	Begin Gr
2	App	Worksheet Menu Bar		10	&Custom	9	
3	App	Worksheet Menu Bar		1	&Show Data Form	1	
4	App	Worksheet Menu Bar		1	&Show Data List	2	
5	App	Worksheet Menu Bar		1	&Print Data List	2	

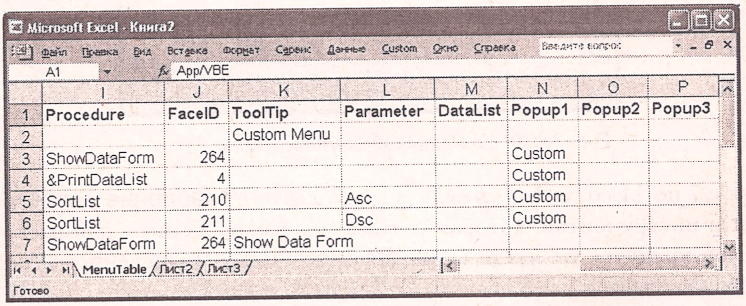
Рис. 26.17. Добавляемые пункты меню

Таблицу можно расширить и добавить новые пункты меню, а также создать командные панели и элементы управления, как показано на рис. 26.18 и рис. 26.19.



	A	B	C	D	E	F	G	H
1	App/VBE	CommandBar	Sub Control ID	Type	Caption	Position	Begin Group	BuiltIn ID
2	App	Worksheet Menu Bar		10	&Custom	9		
3	App	Worksheet Menu Bar		1	&Show Data Form	1		1
4	App	Worksheet Menu Bar		1	&Print Data List	2		1
5	App	Worksheet Menu Bar		1	Sort Names &Ascer	3	TRUE	1
6	App	Worksheet Menu Bar		1	Sort Names &Desce	4		1
7	App	Manage Data		10	&Show Data Form	1		1

Рис. 26.18. Добавление пунктов меню и элементов управления



	I	J	K	L	M	N	O	P
1	Procedure	FaceID	ToolTip	Parameter	DataList	Popup1	Popup2	Popup3
2			Custom Menu					
3	ShowDataForm	264				Custom		
4	&PrintDataList	4				Custom		
5	SortList	210		Asc		Custom		
6	SortList	211		Dsc		Custom		
7	ShowDataForm	264	Show Data Form					

Рис. 26.19. Добавление пунктов меню и элементов управления

Следующие процедуры позволяют автоматизировать запуск этого кода при открытии и закрытии книги:

```
' Подпрограмма: Auto_Open
' Назначение:  Добавление меню и пунктов меню в приложение
Sub Auto_Open()
   SetUpMenus
   CommandBars("Управление данными").Visible = True
End Sub

' Подпрограмма: Auto_Close
' Назначение:  Удаляет меню и пункты меню из приложения
```

```
Sub Auto_Close()
   RemoveMenus
End Sub
```

Данные в строках со 2 по 6 таблицы MenuTable позволяют создавать меню Custom (рис. 26.20), совпадающее с меню Custom, которое создавалось ранее в этой главе (кроме этого, добавлены несколько пиктограмм).

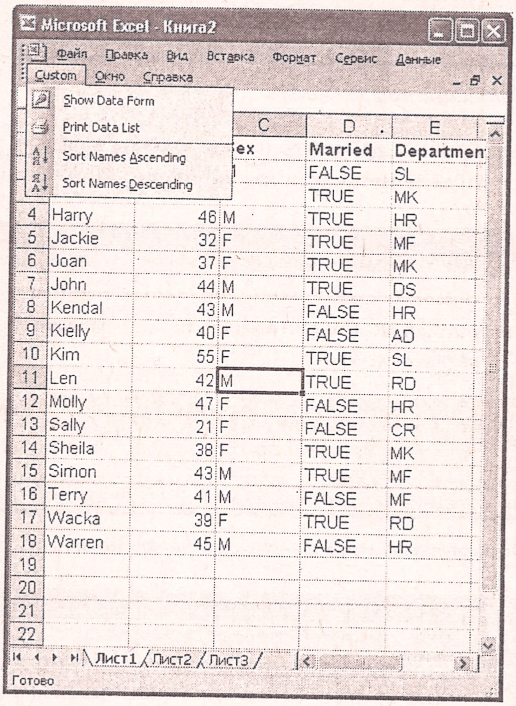


Рис. 26.20. Создание собственного меню

В строках с 7 по 11 создается панель инструментов Управление данными, совпадающая с панелью инструментов, которая создавалась ранее. Данные для раскрывающегося списка отделов хранятся на листе DataLists (рис. 26.21). Выделенному диапазону присвоено имя `Departments`.

Строка 12 описывает пункт Очистить Все (Clear All) в контекстном меню для ячейки листа.

В строке 13 описывается добавление встроенного элемента управления Merge Across в меню Формат⇨Строка (Format⇨Row) (рис. 26.22).

Хотя показанный выше код позволяет добавлять элементы в существующие контекстные меню, он не в состоянии создавать новые контекстные меню. Для этого достаточно будет добавить несколько столбцов в таблицу и соответствующим образом модифицировать код. Эта методика оказывается достаточно гибкой для реализации всех возможных вариантов.

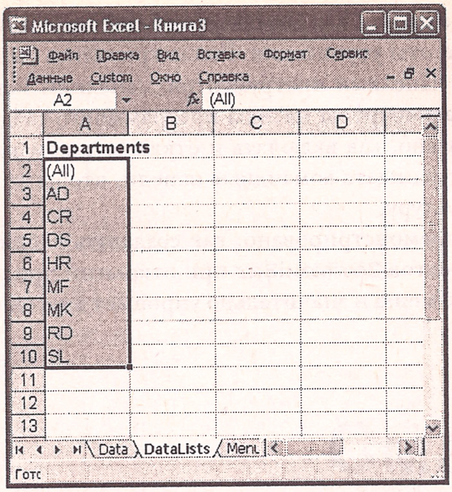


Рис. 26.21. Список с данными для раскрывающегося меню

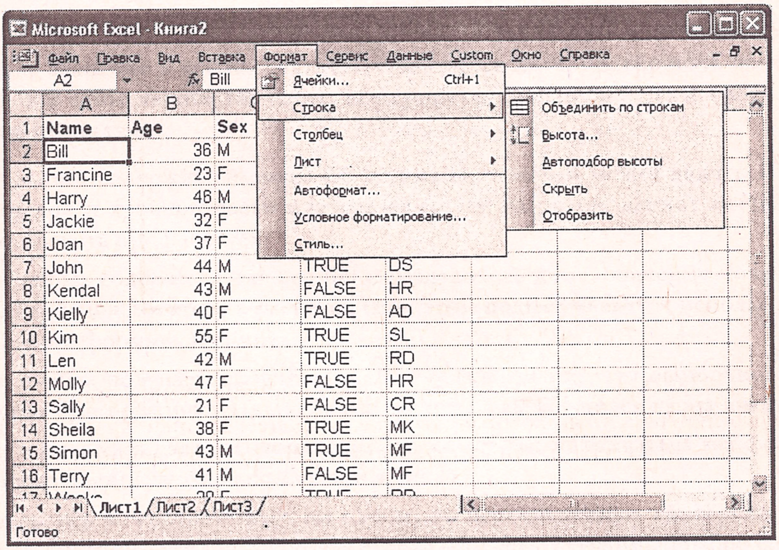


Рис. 26.22. Добавление существующего элемента управления в меню

Резюме

В этой главе рассматривалась структура командных панелей Excel и было показано, как создавать:

- ❑ списки пиктограмм встроенных элементов управления со свойствами `Id` и `FaceId`;
- ❑ полные списки значений `FaceId`, которые доступны в данной версии Excel;
- ❑ полный список элементов всплывающих меню.

Кроме этого, было показано, как создавать собственные командные панели, а также как добавлять в них элементы управления. Рассматривались различия между тремя типами командных панелей: панелями инструментов, панелями меню и всплывающими меню. Кроме того, были рассмотрены методы программного создания этих типов командных панелей. Было показано, как включать и отключать командные панели и элементы управления, а также как защищать командные панели, чтобы пользователи не могли вносить изменения в их структуру.

И наконец, было продемонстрировано, как создавать таблицы с определениями изменений в структуре объектов `CommandBar` при открытии приложения. Этот подход позволяет модифицировать меню и значительно упрощает внесение изменений.

27

Смарт-теги

При вводе значения в ячейку листа Excel пытается интерпретировать введенный набор символов в осмысленное значение. Интерпретация выполняется в соответствии со следующими правилами:

- если запись содержит символы 0-9 и символ, который может являться разделителем целой и дробной части или разделителем разрядов, данное значение можно использовать при расчетах;
- если запись содержит символы 0-9 и разделитель даты, значение может являться датой и использоваться в расчетах (при этом подразумевается использование специального формата ячейки);
- если ячейка получает значение `#N/A`, `#Value!` и т.д., это ошибочное значение, которое будет распространено на все расчеты, ссылающиеся на эту ячейку;
- если запись начинается с символа =, она может рассматриваться как формула, значение которой будет преобразовано в число;
- если запись не соответствует ни одному из перечисленных критериев, это значение рассматривается как текстовая строка — то есть, случайный набор символов.

В любом случае Excel рассматривает введенное значение, как определенный тип данных — число, дату, формулу и т.д. Выбор определенного типа данных подразумевает соответствующее множество типов поведения, операций и свойств (поведение и операции определяются компонентами этого типа). Например, все записи типа "Формула" имеют следующее поведение:

- значение записи проверяется на наличие "синтаксических" ошибок, например, отсутствующие скобки и т.д.;
- записи должны быть включены в дерево зависимости вычислений Excel;

- должно вычисляться значение записи;
- в ячейке должен отображаться результат вычисления, а не текст формулы.

До выхода Excel 2002 правила определения типа ячейки, а также множество типов поведения и свойств были зафиксированы в коде Excel. Единственным способом расширения распознавания типа было предварительное форматирование ячейки (для принудительного определения конкретного типа) или подключение событий `Worksheet_Change`, `Worksheet_SelectionChange` и `Worksheet_Calculate` к соответствующим листам с последующей проверкой введенных, выделенных или измененных данных в процессе пересчета листа. Этот механизм был достаточно громоздким и не позволял просто реагировать на все варианты изменения содержимого ячейки листа.

В Office XP была предоставлена возможность создания смарт-тегов, которая позволяла определять собственные "типы данных" (например, биржевые символы, имена файлов, медицинские термины, номера компонентов и т.д.) и логику распознавания, заставляющую Excel распознавать введенное значение, как принадлежащее определенному типу данных. Кроме того, предоставлялась возможность определить список действий, которые могли выполняться над этим типом данных.

После распознавания записи, как имеющей определенный тип данных, допустимые операции предоставлялись пользователю в виде всплывающего меню при наведении указателя мыши. Например, следующее меню предоставляется для биржевых символов, реализованных с помощью смарт-тегов (рис. 27.1). В данном случае символ MSFT распознается как биржевой символ компании Microsoft.

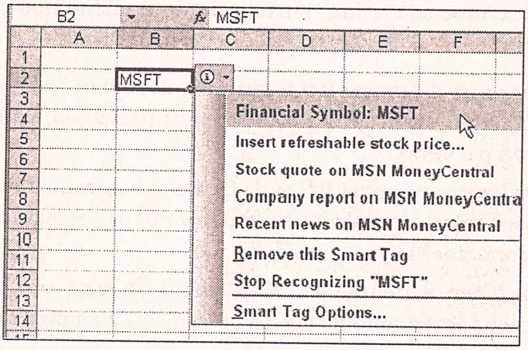


Рис. 27.1. Операции, доступные для типа "биржевой символ"

Для включения смарт-тегов в пределах книги выберите команду Сервис⇨Параметры автозамены (Tools⇨AutoCorrect Options), активизируйте вкладку Смарт-теги (SmartTags), установите флажок Пометить данные смарт-тегами (Label Data with SmartTags) и выберите установленный механизм распознавания типов смарт-тегов.

Компания Microsoft предоставляет на своем сайте официальный инструментарий разработки SmartTag Software Development Kit (SDK) для пакета Office XP. Хотя инструментарий разработки может оказаться полезным, на момент написания книги для пакета Office 2003 такого инструментария разработки еще не существовало. Но в инструментарии разработки для Office XP предоставляется несколько инструментов, которые совместимы с Office 2003 и могут использоваться для определения проблем и получения общего представления о процессе разработки смарт-тегов. Инструментарий доступен на сайте по адресу `http://msdn.microsoft.com/office/`.

Улучшения в механизме смарт-тегов

С выходом пакета Microsoft Office 2003 в механизм смарт-тегов были внесены значительные улучшения, которые реализованы в виде новой библиотеки типов, используемой совместно с библиотекой типов Office XP. В результате и в Office XP, и Office 2003 может применяться один и тот же механизм смарт-тегов. Перед использованием примера реализации механизма смарт-тегов рассмотрим некоторые улучшения, которые стали доступны с выходом Microsoft Office 2003.

Библиотека типов Microsoft SmartTags 2.0

Библиотека типов Microsoft SmartTags 2.0 сохраняет интерфейсы `ISmartTagAciton` и `ISmartTagRecognizer` для обратной совместимости с Office XP. Кроме этих интерфейсов в библиотеке типов версии 2.0 предоставляются интерфейсы `ISmartTagAciton2` и `ISmartTagRecognizer2`. В них предоставляется больше функциональности и поддерживаются возможности смарт-тегов, доступные в пакете Microsoft Office 2003.

Рассмотрим состав каждого интерфейса, а также описания существующих и новых элементов.

- `DisplayPropertyPage` — метод вызывается при щелчке на кнопке **Свойства** (Configure) для настройки объекта `SmartTag`. Метод принимает параметры `SmartTagID` (типа `Long`) и `LocaleID` (типа `Long`). Так как библиотека SmartTag DLL может содержать несколько объектов `SmartTag`, параметр `SmartTagID` позволяет определить отображаемую страницу свойств.
- `PropertyPage` — предназначенное только для чтения свойство, позволяющее приложению определить, поддерживает ли смарт-тег страницы свойств. Как и метод `DisplayPropertyPage`, этот метод принимает параметры `SmartTagID` (типа `Long`) и `LocaleID` (типа `Long`).
- `Recognize2` — метод очень напоминает `Recognize` и принимает несколько дополнительных параметров, позволяющих упростить процесс распознавания текста. Метод `Recognize2` принимает параметр `TokenList` (типа `ISmartTagTokenList`), который разбивает текст на отдельные слова. В результате разработчик освобождается от задачи разбора текста.
- `SmartTagInitialize` — метод вызывается при загрузке смарт-тега и возвращает имя объекта вызывающему приложению.

Интерфейс `ISmartTagAction2` содержит следующие определения:

- `InvokeVerb2` — этот метод выполняет ту же функцию, что и предыдущая версия, которая фактически выполняет операцию при выборе операции над объектом `SmartTag`. Разница между `InvokeVerb2` и `InvokeVerb` заключается в том, что метод `InvokeVerb2` принимает дополнительный параметр `LocaleID`;
- `IsCaptionDynamic` — этот метод определяет некоторые новые возможности объектов `SmartTag` в Office 2003. В пакете Office XP все заголовки смарт-тегов были статическими и не менялись после определения. Метод `IsCaptionDynamic` позволяет определить, действительно ли заголовок является динамическим. Если это так, метод `SmartTagInitialize` вызывается при каждом вызове меню смарт-тега. В результате заголовок можно будет менять на этапе выполнения;

- ShowSmartTagIndicator — этот метод возвращает значение типа Boolean и определяет, выводится ли индикатор смарт-тега;
- SmartTagInitialize — этот метод ничем не отличается от метода SmartTagInitialize из интерфейса ISmartTagRecognizer2, но он вызывается при загрузке компонента Action объекта SmartTag;
- VerbCaptionFromID2 — определяет каждый заголовок в меню действий объекта SmartTag. Этот метод позволяет создавать каскадные меню, использование которых показано на примере приложения.

Если коротко, то новая библиотека типов предоставляет больше информации о региональных параметрах и поддерживает новые возможности, например, каскадные меню, динамические меню и автоматический разбор текста. Теперь пришло время создать смарт-тег, который будет работать в Office XP и Office 2003 и использовать несколько описанных выше новых возможностей.

Смарт-тег FileName

Введите следующий текст в ячейку листа:

```
C:\mydir\mypic.bmp
```

В соответствии с правилами распознавания введенных данных это не число, не дата, не формула, не ошибочное значение и ни один из поддерживаемых в Excel типов. Эта последовательность символов для Excel выглядит случайной. Но для пользователей это — файл, возможно, файл изображения. Это имя файла, принадлежащее типу данных FileName. Если предположить, что эта последовательность символов обозначает имя файла, над именем файла можно выполнить несколько операций:

- проверить существование файла;
- переименовать файл;
- открыть файл в соответствующем приложении;
- если это файл электронной таблицы, открыть его в том же или в новом экземпляре приложения Excel.

Реализация смарт-тега FileName позволит:

- расширить подпрограммы распознавания типов Excel для определения ячеек, содержащих имена файлов;
- предоставить Excel список операций, которые могут выполняться над именами файлов (и над файлами);
- предоставить код для выполнения операций.

> Помните, что в данном случае смарт-тег тестируется в Microsoft Excel, но его точно так же можно использовать в Word, Outlook, PowerPoint и Access. В пакете Microsoft Office XP смарт-теги поддерживались только в Word, Excel и Outlook. В Microsoft Office 2003 эта возможность поддерживается во всех перечисленных ранее приложениях.

Структура объекта SmartTag

Большинство объектов SmartTag состоит из трех компонентов:

- уникального идентификатора типа данных (или нескольких идентификаторов нескольких типов в пределах одного объекта SmartTag);
- модуля класса, который выполняет распознавание, если весь введенный в ячейку текст Excel передает функции этого класса, связывающей ячейку с уникальным идентификатором, и если ячейка проходит условие проверки;
- модуля класса, предоставляющего список операций, доступных для этого типа данных, и выполняющего выбранную пользователем операцию.

Обратите внимание, что можно расширять существующие объекты SmartTag. Для этого необходимо предоставить дополнительные процедуры распознавания, которые будут связывать ячейку с уже существующим идентификатором (например, для расширения списка символов, которые распознаются смарт-тегом Stock Symbol) и/или предоставлять дополнительные операции для существующих идентификаторов (например, для получения биржевой информации не только от MSN, но и от других поставщиков).

Физически объекты SmartTag являются динамически подключаемыми библиотеками ActiveX. Часто такие объекты создаются на языках Visual Basic или C++. Обычно в этой библиотеке хранится класс Recognizer, класс Actions и модуль Globals, в котором хранятся константы и перечисления, используемые обоими классами. Классы Recognizer и Actions должны реализовывать конкретные интерфейсы для предоставления Excel и другим приложениям Microsoft Office возможности вызывать компоненты класса.

Начнем с создания смарт-тега FileName. Откройте Visual Basic 6, создайте новую библиотеку ActiveX DLL и вызовите проект FileNameSmartTag. Интерфейсы Recognizer и Actions определяются в библиотеке типов "Microsoft Smart Tags 2.0 Type Library". Ссылку на эту библиотеку необходимо создать в диалоговом окне Проект⇨Ссылки (Tools⇨References). Для этого напротив названия библиотеки нужно установить флажок. Эти интерфейсы предназначены для обратной совместимости с Microsoft Office XP.

Уникальный идентификатор смарт-тега

При связывании ячейки листа с объектом SmartTag ссылка хранится в книге в формате XML. Это значит, что идентификатор SmartTag должен создаваться специальным образом — в виде указателя Uniform Resource Identifier (URI), после которого указывается символ '#' и имя конкретного типа объекта SmartTag. Строка URI предоставляет возможность идентифицировать источник идентификатора и группы ID. Комбинация идентификатора и группы позволяет получить глобально уникальный и однозначный идентификатор. Полная строка идентификатора конкретного типа данных имеет следующий формат:

```
urn:schemas-<company>-com:<group>#SmartTagName
```

В данном примере распознается три типа имени файла:

- файлы, которые можно открыть в Excel, получают идентификатор "urn:schemas-wrox-com:Excel2003ProgRef#ExcelFile";

- файлы, которые можно открыть в Word, получают идентификатор "urn:schemas-wrox-com:Excel2003ProgRef#WordFile";
- файлы, которые можно открыть в других приложениях, получают идентификатор "urn:schemas-wrox-com:Excel2003ProgRef#NonHostFile".

Добавьте в проект Visual Basic новый модуль Globals и эти идентификаторы в виде констант.

```
'В этом модуле хранятся глобальные константы и перечисления,
' которые используются объектами SmartTag

' Универсальные идентификаторы ресурсов для типов данных SmartTag
Public Const psURIExcel =
↳ "urn:schemas-wrox-com:Excel2003ProgRef#ExcelFile"
Public Const psURIWord =
↳ "urn:schemas-wrox-com:Excel2003ProgRef#WordFile"
Public Const psURINonHost = "urn:schemas-wrox-
↳com:Excel2003ProgRef#NonHostFile"
```

Многие создаваемые здесь процедуры объектов SmartTag вызываются несколько раз для перебора последовательности, например имен Recognizer или действий SmartTag, с передачей номера последовательности в качестве параметра. Для улучшения читаемости желательно определять соответствующие номерам последовательностей перечисления (Enumerations или Enums) в модуле Globals. Следующие перечисления используются далее в этой главе, но определяются здесь, в модуле Globals.

```
' Номера последовательностей для типов данных SmartTag
Public Enum peTagSequence
   peTagExcel = 1
   peTagWord = 2
   peTagNonHost = 3
End Enum

' Идентификаторы выполняемых операций
Public Enum peFileVerbs

   ' Четыре глагола для файлов Excel
   peXLExists = 11
   peXLRename = 12
   peXLOpen = 13
   peXLOpenNew = 14

   ' Четыре глагола для файлов Word
   peWordExists = 21
   peWordRename = 22
   peWordOpen = 23
   peWordOpenNew = 24

   ' Три глагола для других файлов
   peNHExists = 31
   peNHRename = 32
   peNHOpen = 33
End Enum
```

Класс Recognizer

Взаимодействие между Excel (или другим приложением Microsoft Office) и процедурой распознавания смарт-тега осуществляется через специальный интерфейс — `ISmartTagRecognizer`, который реализован в классе.

Для создания класса, реализующего интерфейс, переименуйте принятый по умолчанию класс `Class1` в `Recognizer` и введите в класс следующий код:

```
' Модуль распознавания имен файлов
Option Explicit
Implements ISmartTagRecognizer
```

Этот интерфейс предоставляет следующие методы и свойства. Все они должны быть реализованы в классе.

Методы и свойства	*Описание*
`ProgID`	Идентификатор класса `Recognizer`, который определяется, как `<Название_проекта>.<Имя_класса>`
`Name`	Короткое имя объекта `SmartTag`, которое отображается в списке объектов `SmartTag`
`Desc`	Описание объекта `SmartTag`
`SmartTagCount`	Количество типов данных `SmartTag`, которые распознаются этим классом
`SmartTagName`	Вызывается по одному разу для каждого класса для предоставления уникального идентификатора типа данных
`SmartTagDownloadURL`	Если файл открывается на компьютере, на котором не установлена библиотека `SmartTag`, значение этого свойства указывает на источник загрузки библиотеки `Actions`.
`Recognize`	Основная подпрограмма, которая выполняет распознавание и обработку текста

Первые три компонента одинаковы для всех объектов `SmartTag`.

```
Private Property Get ISmartTagRecognizer_ProgId() As String
  ISmartTagRecognizer_ProgId = "FileNameSmartTag.Recognizer"
End Property

Private Property Get ISmartTagRecognizer_Name( _
                        ByVal LocaleID As Long) As String
   ISmartTagRecognizer_Name = "Filename SmartTag Recognizer"
End Property

Private Property Get ISmartTagRecognizer_Desc( _
                        ByVal LocaleID As Long) As String
   ISmartTagRecognizer_Desc = "SmartTag DLL to recognize filenames"
End Property
```

Свойство `SmartTagCount` просто возвращает количество распознаваемых классом типов данных. В данном случае распознается три типа данных — файлы Excel, файлы Word и файлы других форматов.

```
Private Property Get ISmartTagRecognizer_SmartTagCount() As Long
   ISmartTagRecognizer_SmartTagCount = 3
End Property
```

Свойство SmartTagName вызывается несколько раз по разу для каждого распознаваемого типа данных. В качестве параметра свойству передается номер типа данных (в данном случае 1, 2 или 3). Свойство возвращает уникальный идентификатор (URI) для данного типа данных. Многие методы интерфейса SmartTag применяют эту технику передачи номера для идентификации конкретного значения или операции. Для улучшения читаемости кода можно использовать перечислимый тип, значения которого соответствуют трем последовательным номерам; перечислимый тип peTagSequence определен в показанном ранее модуле Globals. Значения этого перечислимого типа используются свойством SmartTagName.

```
Private Property Get ISmartTagRecognizer_SmartTagName( _
                        ByVal SmartTagID As Long) As String

   ' Вернуть идентификатор URI для указанного
   ' последовательного номера
   Select Case SmartTagID

      Case peTagExcel
         ISmartTagRecognizer_SmartTagName = psURIExcel

      Case peTagWord
         ISmartTagRecognizer_SmartTagName = psURIWord

      Case peTagNonHost
         ISmartTagRecognizer_SmartTagName = psURINonHost
   End Select
End Property
```

В свойстве SmartTagDownloadURL подобный код можно применять для предоставления ссылки на библиотеку Actions. В данном случае оба класса Action и Recognizer расположены в одной библиотеке DLL.

```
Private Property Get ISmartTagRecognizer_SmartTagDownloadURL( _
                        ByVal SmartTagID As Long) As String
   ISmartTagRecognizer_SmartTagDownloadURL = ""
End Property
```

Класс Recognizer выполняет все необходимые операции и является самым сложным компонентом механизма SmartTag. Правильный класс Recognizer должен:

- распознавать правильные последовательности символов;
- не давать ложных срабатываний;
- работать быстро.

Если необходимо распознавать элементы списка терминов (например, номера компонентов), желательно загрузить полный список элементов в массив (это можно сделать в одном из свойств, например ProgID), особенно если список придется получать с другого Web-сайта или извлекать из базы данных.

Несмотря на качество механизма распознавания, скорее всего, будут возникать ложные срабатывания. Например, последовательность символов "TRUE" одновременно может соответствовать значению типа Boolean и действительному биржевому символу. В результате ячейки со значением True типа Boolean будут распознаваться как бирже-

вые символы. Точно так же процедура распознавания моделей автомобилей может распознать числа 106, 206, 405 и т.д. как действительные номера моделей, хотя на самом деле это результаты вычислений в ячейках Excel.

В данном случае для распознавания строки как имени файла, строка должна соответствовать одному из следующих условий:

- строка может иметь формат `<буква>:\<текст>.<расширение>` без пробела после точки;
- строка может иметь формат `\\<текст>\<текст>.<расширение>`.

В результате некоторые имена файлов не будут распознаны (например, не все имена файлов не имеют расширения), но при этом будут распознаны некоторые строки, которые не являются именами файлов. Для улучшения алгоритма можно проверять наличие недопустимых символов внутри имени файла, но даже приведенные выше правила являются хорошим примером.

Кроме этого, файлы поддерживаемых форматов имеют расширение `.xls` и `.xla` в Excel и `.doc` и `.dot` в Word (в зависимости от приложения). Следующий код добавляется в класс `Recognizer`:

```
Private Sub ISmartTagRecognizer_Recognize(ByVal Text As String, _
               ByVal DataType As SmartTagLib.IF_TYPE, _
               ByVal LocaleID As Long, _
               ByVal RecognizerSite As _
                     SmartTagLib.ISmartTagRecognizerSite)

   ' Переменные, которые используются для поиска имени файла в тексте
   Dim iStart As Long, iEnd As Long, iDot As Long, iSlash As Long

   ' Переменная, в которой хранится имя файла
   Dim sExt As String

   ' Переменная, в которой хранятся свойства объекта SmartTag
   Dim oPropBag As SmartTagLib.ISmartTagProperties

   iEnd = 1

   ' Абзац Word может содержать несколько имен файлов,
   ' которые придется перебрать по порядку
   Do
   iDot = 0

   ' Найти символы в начале имени файла
   iStart = InStr(iEnd, Text, ":\")
   If iStart > 0 Then
      iStart = iStart - 1
      iDot = InStr(iStart, Text, ".")
   Else
      iStart = InStr(iEnd, Text, "\\")
      iSlash = InStr(iStart, Text, "\")

      If iSlash > 0 Then
         iDot = InStr(iSlash, Text, ".")
      End If
   End If

   ' Если найдено начало и конец имени файла...
   If iStart > 0 And iDot > 0 Then
```

```
      '... проверить наличие символа сразу после точки
      If Trim$(Mid$(Text, iDot + 1, 1)) <> "" Then
         ' Действительное имя файла!

         ' Найти конец расширения, после которого может
         ' существовать другой текст, если речь идет об абзаце Word
         For iEnd = iDot + 1 To Len(Text)
            If InStr(1, "\:*?""<>| ", Mid$(Text, iEnd, 1)) _
                          <> 0 Then Exit For
         Next

         ' Получить расширение
         sExt = Mid$(Text, iDot + 1, iEnd - iDot)

         ' Получить свойства этого объекта SmartTag
         Set oPropBag = RecognizerSite.GetNewPropertyBag

         ' Добавить имя файла и расширение в свойства объекта
         ' для последующего использования
         oPropBag.Write "FileName", Mid$(Text, iStart, iEnd - iStart)
         oPropBag.Write "Extension", sExt

         ' Соответствует ли имя поддерживаемому типу файла
         ' В Excel используется тип данных IF_TYPE_CELL, а в Word
         ' используется тип данных IF_TYPE_PARA.
         ' Фактическое распознавание текста выполняется в методе
         ' CommitSmartTag.
         If (DataType = IF_TYPE_CELL And _
               (sExt = "xls" Or sExt = "xla")) Then

            RecognizerSite.CommitSmartTag psURIExcel, _
                          iStart, iEnd - iStart, oPropBag
            ElseIf (DataType = IF_TYPE_PARA And _
                   (sExt = "doc" Or sExt = "dot")) Then

               RecognizerSite.CommitSmartTag psURIWord, _
                          iStart, iEnd - iStart, oPropBag
            Else
               RecognizerSite.CommitSmartTag psURINonHost, _
                          iStart, iEnd - iStart, oPropBag
            End If
         End If
      End If
   Loop Until iDot = 0 Or iEnd >= Len(Text)

End Sub
```

Класс Actions

После распознавания компонента с помощью класса Recognizer, Excel должна найти элементы для отображения во всплывающем меню объекта SmartTag. Эти элементы соответствуют операциям, поддерживаемым этим типом данных смарт-тега. При выборе одной из этих операций Excel должна обратиться к библиотеке SmartTag и выполнить запрошенную операцию. Все эти действия выполняются через интерфейс ISmartTagAction.

Для создания класса, реализующего данный интерфейс, добавьте в проект новый модуль класса, выберите режим создания экземпляров '5 - MultiUse', переименуйте класс в Actions и введите следующий код:

```
'Модуль для операций объекта SmartTag для имен файлов

Option Explicit
Implements ISmartTagAction
```

Интерфейс `ISmartTagAction` предоставляет следующие методы и свойства. Все они должны быть реализованы.

Методы и свойства	***Описание***
`ProgID`	Идентификатор класса `Recognizer`. Определяется как `<ProjectName>.<ClassName>`
`Name`	Короткое имя объекта `SmartTag`, которое отображается в списке объектов
`Desc`	Описание объекта `SmartTag`
`SmartTagCount`	Количество типов данных `SmartTag`, распознаваемых классом
`SmartTagName`	Вызывается по одному разу для каждого типа данных для предоставления уникального идентификатора типа данных
`SmartTagCaption`	Вызывается по одному разу для каждого типа данных и передает заголовок, который выводится во всплывающем меню объекта `SmartTag`
`VerbCount`	Вызывается для каждого типа данных и передает количество допустимых операций
`VerbID`	Вызывается для каждой комбинации типа данных и операции. Предоставляет уникальный идентификатор этой комбинации
`VerbCaptionFromID`	Вызывается для каждого идентификатора операции. Предоставляет заголовок для вывода во всплывающем меню объекта `SmartTag`
`VerbNameFromID`	Вызывается для каждого идентификатора операции. Предоставляет программное имя операции (такие имена позволяют вызывать операции из кода VBA — дополнительная информация приводится далее в этой главе)
`InvokeVerb`	Вызывается при щелчке на одном из меню объекта `SmartTag` или при вызове операции из кода VBA. Это основная подпрограмма, выполняющая выбранную операцию

Первые пять свойств не отличаются от свойств интерфейса `ISmartTagRecognizer`.

```
Private Property Get ISmartTagAction_ProgId() As String
   ISmartTagAction_ProgId = "FileNameSmartTag.Actions"
End Property

Private Property Get ISmartTagAction_Name(ByVal LocaleID As Long)
↳ As String
   ISmartTagAction_Name = "Filename SmartTag Actions"
End Property

Private Property Get ISmartTagAction_Desc(ByVal LocaleID As Long)
↳ As String
   ISmartTagAction_Desc = "Provides actions to perform on filenames"
End Property

Private Property Get ISmartTagAction_SmartTagCount() As Long
   ISmartTagAction_SmartTagCount = 3
End Property

Private Property Get ISmartTagAction_SmartTagName( _
```

```
                          ByVal SmartTagID As Long) As String

    ' Вернуть интересующий идентификатор URI для номера
    ' объекта SmartTag
    Select Case SmartTagID
        Case peTagExcel
            ISmartTagAction_SmartTagName = psURIExcel

        Case peTagWord
            ISmartTagAction_SmartTagName = psURIWord

        Case peTagNonHost
            ISmartTagAction_SmartTagName = psURINonHost
    End Select

End Property
```

Свойство SmartTagCaption предоставляет заголовок, который отображается во всплывающем меню объекта SmartTag:

```
Private Property Get ISmartTagAction_SmartTagCaption( _
            ByVal SmartTagID As Long, ByVal LocaleID As Long)
↳ As String

    ' Вернуть заголовок объекта SmartTag
    Select Case SmartTagID
        Case peTagExcel
            ISmartTagAction_SmartTagCaption = "Файлы Excel"

        Case peTagWord
            ISmartTagAction_SmartTagCaption = "Файлы Word"

        Case peTagNonHost
            ISmartTagAction_SmartTagCaption = "Имена файлов"
    End Select

End Property
```

Свойство VerbCount возвращает количество операций для каждого объекта SmartTag. В данном примере предоставляются следующие операции:

- проверка существования файла;
- переименование файла (также в результате этой операции обновляется содержимое документа);
- открытие файла в принятом по умолчанию приложении;
- если Excel или Word распознали собственный файл, предоставляется дополнительная операция по открытию файла в новом экземпляре Excel или Word.

Для большей читаемости кода для этих операций в модуле Globals определено перечисление peFileVerbs, в котором указаны идентификаторы, соответствующие идентификаторам объектов SmartTag и операций.

Свойство VerbCount класса Actions используется для получения количества операций, доступных для каждого типа данных.

```
Private Property Get ISmartTagAction_VerbCount( _
                    ByVal SmartTagName As String) As Long

    Select Case SmartTagName
        Case psURIExcel
```

```
         ISmartTagAction_VerbCount = 4

      Case psURIWord
         ISmartTagAction_VerbCount = 4

      Case psURINonHost
         ISmartTagAction_VerbCount = 3
   End Select

End Property
```

Свойство VerbID используется для получения уникального идентификатора каждой операции. Точная нумерация операций в перечислении peFileVerbs позволяет вычислить идентификатор на основе номеров объектов SmartTag и номеров операций:

```
Private Property Get ISmartTagAction_VerbID( _
                        ByVal SmartTagName As String, _
                        ByVal VerbIndex As Long) As Long

   Select Case SmartTagName
      Case psURIExcel
         ISmartTagAction_VerbID = 10 + VerbIndex

      Case psURIWord
         ISmartTagAction_VerbID = 20 + VerbIndex

      Case psURINonHost
         ISmartTagAction_VerbID = 30 + VerbIndex
   End Select

End Property
```

Теперь, когда программа Excel получила информацию о всех предоставляемых операциях, каждой операции присваивается заголовок, отображаемый во всплывающем меню объекта SmartTag.

```
Private Property Get ISmartTagAction_VerbCaptionFromID( _
      ByVal VerbID As Long, ByVal ApplicationName As String, _
      ByVal LocaleID As Long) As String

  Select Case VerbID
    Case peXLExists, peWordExists, peNHExists
    ISmartTagAction_VerbCaptionFromID = "Проверить
↳ существование файла"

    Case peXLRename, peWordRename, peNHRename
    ISmartTagAction_VerbCaptionFromID = "Переименовать файл"

    Case peXLOpen, peWordOpen, peNHOpen
    ISmartTagAction_VerbCaptionFromID = "Открыть файл"

    Case peXLOpenNew
    ISmartTagAction_VerbCaptionFromID = _
         "Открыть файл в новом экземпляре Excel"

    Case peWordOpenNew
    ISmartTagAction_VerbCaptionFromID = _
         "Открыть файл в новом экземпляре Word"

   End Select

End Property
```

Excel предоставляет механизм включения операций SmartTag из кода VBA. Свойство VerbNameFromID используется для предоставления программного имени каждой операции. Далее в этой главе приводится дополнительная информация о применении кода VBA для управления объектами SmartTag и вызова их операций в Excel.

```
Private Property Get ISmartTagAction_VerbNameFromID( _
                        ByVal VerbID As Long) As String

   Select Case VerbID
      Case peXLExists, peWordExists, peNHExists
         ISmartTagAction_VerbNameFromID = "CheckExists"

      Case peXLRename, peWordRename, peNHRename
         ISmartTagAction_VerbNameFromID = "Rename"

      Case peXLOpen, peWordOpen, peNHOpen
         ISmartTagAction_VerbNameFromID = "Open"

      Case peXLOpenNew, peWordOpenNew
         ISmartTagAction_VerbNameFromID = "OpenNew"

   End Select

End Property
```

Главной подпрограммой класса Actions является метод InvokeVerb. В этом методе выполняется выбранная операция. Обратите внимание, что в качестве одного из параметров методу передается объект Target. При вызове из Excel значением параметра является объект Range, в котором хранится распознаваемый текст. Это позволяет получить полный доступ для запроса и модификации любого объекта Excel. В данном случае текст имени файла будет модифицироваться при вызове операции Rename.

Функция Windows API ShellExecute позволяет открывать файлы в соответствующем приложении (дополнительная информация о Windows API приводится в главе 16). Следующее определение необходимо добавить в начале класса Actions:

```
' Для открытия файла используется вызов API ShellExecute
Private Declare Function ShellExecute Lib "shell32.dll" _
        Alias "ShellExecuteA" _
        (ByVal hwnd As Long, ByVal lpOperation As String, _
        ByVal lpFile As String, ByVal lpParameters As String, _
        ByVal lpDirectory As String, ByVal nShowCmd As Long) As Long
```

Для выполнения запрошенной операции используется нижеприведенный код:

```
Private Sub ISmartTagAction_InvokeVerb(ByVal VerbID As Long, _
        ByVal ApplicationName As String, ByVal Target As Object, _
        ByVal Properties As SmartTagLib.ISmartTagProperties, _
        ByVal Text As String, ByVal Xml As String)

   Dim sNewName As String
   Dim oHost As Object
   Dim a As Long

   ' Все операции требуют существования файла, поэтому эта
   ' проверка выполняется в первую очередь
   If Not FileExists(Text) Then
      MsgBox "Файл '" & Text & "' не существует", _
             vbOKOnly + vbCritical, "FileName Smart Tag"
   Else
```

```
Select Case VerbID
   Case peXLExists, peWordExists, peNHExists

   ' Если условие выполнено, файл существует
   MsgBox "The file '" & Text & "' exists", _
         vbOKOnly, "FileName Smart Tag"

   Case peXLRename, peWordRename, peNHRename

      ' Получить новое имя файла, передав исходное
      ' имя файла в качестве принятого по умолчанию значения
      sNewName = InputBox("Введите новое имя файла", _
                        "FileName Smart Tag", Text)

      ' Если имя изменилось (и пользователь не щелкнул
      ' на кнопке Отмена)...
      If sNewName <> "" And sNewName <> Text Then
         On Error Resume Next
            '... переименовать файл
         Name Text As sNewName
            If Err = 0 Then
            ' Файл успешно переименован, поэтому можно
            ' изменить строку имени файла
            ' Каждое приложение  (Excel, Word или IE)
            ' требует использования собственного синтаксиса:
             Select Case Left$(LCase$(ApplicationName), 5)
             Case "excel"
                Target.Value = sNewName
             Case "word."
                Target.Text = sNewName
             Case Else
                Target.InnerText = sNewName
            End Select
         End If
      End If

   Case peXLOpen
      ' Открыть файл в Excel
      Target.Application.Workbooks.Open Text

   Case peWordOpen
      ' Открыть файл в Word
      Target.Application.documents.Open Text

   Case peNHOpen
      ' Воспользоваться функцией API ShellExecute
      ' для открытия файла в принятом по умолчанию редакторе
      On Error Resume Next
      a = ShellExecute(0, vbNullString, Text, vbNullString, _
                  vbNullString, 1)

   Case peXLOpenNew
      ' Открыть файл Excel в новом экземпляре Excel
      Set oHost = CreateObject("Excel.Application")
      oHost.Workbooks.Open Text
      oHost.Visible = True

   Case peWordOpenNew
      ' Открыть файл Word в новом экземпляре Word
      Set oHost = CreateObject("Word.Application")
      oHost.documents.Open Text
```

```
            oHost.Visible = True

      End Select
   End If
End Sub
```

В подпрограмме InvokeVerb для проверки существования файла используется отдельная функция:

```
Private Function FileExists(sFile As String) As Boolean

   Dim sDir As String

   On Error Resume Next
   sDir = Dir(sFile)
   FileExists = (sDir <> "")

End Function
```

На этом реализация завершается. Сохраните проект Visual Basic и создайте библиотеку DLL.

Реализация возможностей объектов SmartTag в Office 2003

К этому моменту был создан полностью функциональный смарт-тег, который может работать как в Microsoft Office XP, так и в Microsoft Office 2003. Теперь в каждый класс будет добавлено несколько строк кода, которые позволят использовать возможности Office 2003. В класс Recognizer необходимо добавить выделенный ниже код, что позволит реализовать интерфейс ISmartTagRecognizer2.

```
' Модуль для распознавания имен файлов
Option Explicit
Implements ISmartTagRecognizer
Implements ISmartTagRecognizer2
Private Property Get ISmartTagRecognizer_ProgId() As String
   ISmartTagRecognizer_ProgId = "FilenameSmartTag.Recognizer"
End Property
```

Как было показано ранее в этой главе, этот метод требует реализации только четырех методов, поэтому после кода приводится краткое описание принципов его работы. Добавьте следующий код в конец класса Recognizer:

```
' Методы интерфейса ISmartTagRecognizer2
Public Sub ISmartTagRecognizer2_Recognize2(ByVal Text As String, _
     ByVal DataType As SmartTagLib.IF_TYPE, _
     ByVal LocaleID As Long, _
     ByVal RecognizerSite2 As SmartTagLib.ISmartTagRecognizerSite2, _
     ByVal ApplicationName As String,
     ByVal TokenList As SmartTagLib.ISmartTagTokenList)

     ISmartTagRecognizer_Recognize Text, _
     DataType, LocaleID, RecognizerSite2
End Sub
```

В методе Recognize2 можно воспользоваться средствами разбора текста на компоненты, но для сохранения ясности используется вызов исходного метода Recognizer, чтобы Office XP и Office 2003 применяли один и тот же метод для определения соответствия текста требованиям объекта SmartTag. Добавьте в класс следующие три метода:

```
Public Property Get ISmartTagRecognizer2_PropertyPage
        (ByVal SmartTagID As Long, _
         ByVal LocaleID As Long) As Boolean

    ISmartTagRecognizer2_PropertyPage = False
End Property

Public Sub ISmartTagRecognizer2_DisplayPropertyPage( _
        ByVal SmartTagID As Long, _
        ByVal LocaleID As Long)
End Sub

Public Sub ISmartTagRecognizer2_SmartTagInitialize( _
        ByVal AppName As String)
End Sub
```

Метод PropertyPage возвращает значение False, так как страница со свойствами этого компонента не предоставляется. Если бы страница со свойствами предоставлялась, она выводилась бы методом DisplayPropertyPage. К этому моменту все необходимые методы интерфейса ISmartTagRecognizer2 уже реализованы.

Теперь для реализации интерфейса ISmartTagAction2 необходимо внести следующие изменения в класс Actions:

```
Option Explicit
Implements ISmartTagAction
Implements ISmartTagAction2
Private Property Get ISmartTagAction_ProgId() As String
   ISmartTagAction_ProgId = "FilenameSmartTag.Actions"
End Property
```

Метод InvokeVerb2 отвечает за передачу всей необходимой информации исходному методу InvokeVerb. Добавьте следующий код в класс Actions:

```
Public Sub ISmartTagAction2_InvokeVerb2(ByVal VerbID As Long, ByVal _
      ApplicationName As String, ByVal Target As Object, ByVal _
      Properties As SmartTagLib.ISmartTagProperties, ByVal _
      Text As String, ByVal Xml As String, ByVal LocaleID As Long)

ISmartTagAction_InvokeVerb VerbID, ApplicationName, _
Target, Properties, Text, Xml

End Sub
```

В данном примере динамические заголовки не отображаются. Кроме того, в объектах SmartTag должны использоваться индикаторы подчеркивания, которые реализуются с помощью показанного ниже кода. Добавьте следующий код в конец классаActions:

```
Public Property Get ISmartTagAction2_IsCaptionDynamic(ByVal _
       VerbID As Long, ByVal ApplicationName As String, ByVal _
       LocaleID As Long) As Boolean

    ISmartTagAction2_IsCaptionDynamic = False
End Property

Public Property Get ISmartTagAction2_ShowSmartTagIndicator(ByVal _
       VerbID As Long, ByVal ApplicationName As String, ByVal _
       LocaleID As Long) As Boolean

    ISmartTagAction2_ShowSmartTagIndicator = True
End Property
```

```
Public Sub ISmartTagAction2_SmartTagInitialize(ByVal _
      ApplicationName As String)
End Sub
```

Код для использования расширенных возможностей Office 2003 практически завершен. Осталось добавить метод VerbCaptionFromID2, фактически предоставляющий текст для всплывающих меню смарт-тега. Office 2003 поддерживает подменю, поэтому здесь используется код метода VerbCaptionFromID с минимальными модификациями.

```
Public Property Get ISmartTagAction2_VerbCaptionFromID2(ByVal _
      VerbID As Long, ByVal ApplicationName As String, ByVal _
      LocaleID As Long, ByVal _
      Properties As SmartTagLib.ISmartTagProperties, ByVal _
      recognized_text As String, ByVal Xml As String, ByVal _
      Target As Object) As String

   Select Case VerbID
      Case peXLExists, peWordExists, peNHExists
         ISmartTagAction2_VerbCaptionFromID2 = "Проверить
↳ существование файла"

      Case peXLRename, peWordRename, peNHRename
         ISmartTagAction2_VerbCaptionFromID2 = "Переименовать файл"

      Case peXLOpen, peWordOpen, peNHOpen
         ISmartTagAction2_VerbCaptionFromID2 = "Открыть///Файл"

      Case peXLOpenNew
         ISmartTagAction2_VerbCaptionFromID2 = _
                          "Открыть///В новом экземпляре Excel"

      Case peWordOpenNew
         ISmartTagAction2_VerbCaptionFromID2 = _
                 "Открыть///Открыть файл в новом экземпляре Word"

   End Select
End Property
```

Глядя на код, легко заметить, что он ничем не отличается от кода метода VerbCaptionFromID, кроме трех пунктов меню, содержащих символы косой черты (///). Эти символы указывают на использование подменю. В данном случае главное меню будет называться Открыть (Open), а три пункта меню будут являться пунктами подменю. Эта возможность доступна только при работе в Microsoft Office 2003.

Регистрация смарт-тега

Как и при использовании надстроек COM, пакет Office должен знать о существовании библиотеки SmartTag. Для этого в системный реестр Windows необходимо внести соответствующие записи. В отличие от надстроек COM компания Microsoft не предоставила необходимых инструментов для упрощения процедуры регистрации. Для этого придется вручную создавать необходимые разделы системного реестра. Для создания записей можно воспользоваться файлом .reg, содержимое которого показано ниже.

При запуске Excel сканирует системный реестр в поисках следующего раздела:

```
HKEY_CURRENT_USER\Software\Microsoft\Office\Common\Smart
↳ Tag\Recognizers
```

Обычно имя такого раздела соответствует идентификатору ProgID или идентификатору ClassID класса Recognizer для объектов SmartTag. Полученный идентификатор применяется для создания экземпляра класса. Для получения остальной информации об объекте SmartTag используются свойства интерфейса ISmartTagRecognizer. Этот процесс повторяется и для классов Actions.

Сначала необходимо определить идентификаторы классов, сгенерированные средой разработки Visual Basic. Для этого выберите команду Пуск⇨Выполнить (Start⇨Run) и запустите приложение regedit. В списке слева найдите раздел HKEY_CLASSES_ROOT\ <Имя_проекта>.<Имя_класса>\Clsid и обратите внимание на значение записи (Default). Это значение соответствует идентификатору класса. Для копирования идентификатора выполните двойной щелчок на записи (Default) и воспользуйтесь комбинацией клавиш <Ctrl+C> для копирования значения в буфер обмена.

Для примера FileName раздел системного реестра Actions называется HKEY_CLASSES_ROOT\FileNameSmartTag.Actions\Clsid.

В данном случае идентификатор класса Actions выглядит как {76C5099E-36E2-4BBE-BA02-6097FAF5CFA2}.

Для класса Recognizer раздел системного реестра называется HKEY_CLASSES_ROOT\ FileNameSmartTag.Recognizer\Clsid.

Идентификатор класса Recognizer выглядит как {76C5099E-36E2-4BBE-BA02-6097FAF5CFA2}.

Эти идентификаторы классов используются для регистрации библиотеки SmartTag (идентификаторы необходимо указать в классе .reg). Для этого откройте редактор Блокнот (Notepad) и скопируйте в него следующий текст. Показанные выше идентификаторы необходимо скопировать вместо строк ActionClassID и RecognizerClassID:

```
REGEDIT4

[HKEY_CURRENT_USER\Software\Microsoft\Office\Common\Smart Tag]

[HKEY_CURRENT_USER\Software\Microsoft\Office\Common\Smart
↳ Tag\Actions]
[HKEY_CURRENT_USER\Software\Microsoft\Office\Common\Smart
↳ Tag\Actions \ActionClassID]

[HKEY_CURRENT_USER\Software\Microsoft\Office\Common\Smart
↳ Tag\Recognizers]

[HKEY_CURRENT_USER\Software\Microsoft\Office\Common\Smart
↳ Tag\Recognizers\RecognizerClassID]
```

Удалите пустые строки в начале файла и сохраните файл с расширением .reg.

Полученный в результате файл FileNameSmartTag.reg состоит из шести строк и выглядит следующим образом:

```
REGEDIT4

[HKEY_CURRENT_USER\Software\Microsoft\Office\Common\Smart Tag]

[HKEY_CURRENT_USER\Software\Microsoft\Office\Common\Smart
↳ Tag\Actions]

[HKEY_CURRENT_USER\Software\Microsoft\Office\Common\Smart
↳Tag\Actions\{76C5099E-36E2-4BBE-BA02-6097FAF5CFA2}]
```

```
[HKEY_CURRENT_USER\Software\Microsoft\Office\Common\Smart
➥Tag\Recognizers]

[HKEY_CURRENT_USER\Software\Microsoft\Office\Common\Smart
➥Tag\Recognizers\{76C5099E-36E2-4BBE-BA02-6097FAF5CFA2}]
```

Выполните двойной щелчок на файле `.reg` в окне Проводника Windows (Windows Explorer), и соответствующие записи будут добавлены в системный реестр. После этого Excel будет распознавать смарт-тег `FileName`.

Использование смарт-тега FileName

Запустите Excel 2002, выберите команду Сервис⇨Параметры автозамены (Tools⇨AutoCorrect Options) и активизируйте вкладку Смарт-теги. Для включения объектов `SmartTag` установите флажок Пометить данные смарт-тегами (Label data with SmartTags). В списке должна присутствовать запись "Filename SmartTag Recognizer". Установите соответствующий флажок и щелкните на кнопке OK. Ниже показано диалоговое окно (рис. 27.2), в котором установлены все необходимые флажки.

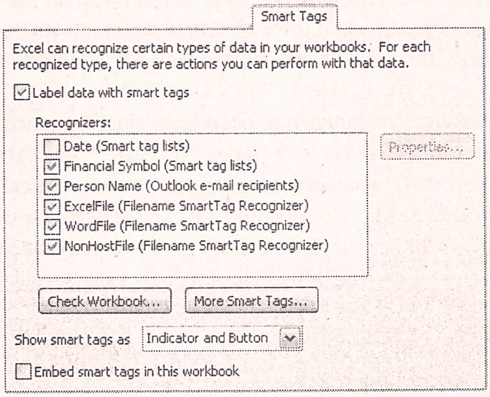


Рис. 27.2. Включение смарт-тегов FileName

Все ячейки, содержимое которых соответствует критериям имени файла, получат индикатор смарт-тега (темный треугольник в нижнем правом углу). При щелчке на символе `i` станет доступно меню с операциями над смарт-тегом `FileName` (рис. 27.3).

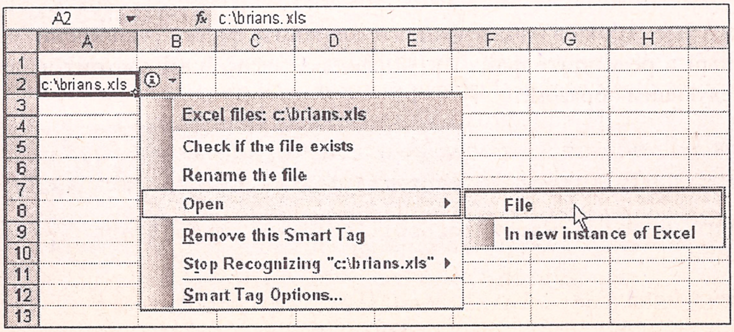


Рис. 27.3. Меню операций над смарт-тегом

Управление смарт-тегами из кода VBA

Так исторически сложилось, что компания Microsoft обеспечивает управление всеми возможностями Excel с помощью языка VBA. Смарт-теги не являются исключением. Следующие объекты и свойства позволяют реализовать в коде те же операции, которые доступны через пользовательский интерфейс.

Объект/Свойство	*Описание*
`Workbook.SmartTagOptions`	Объект, через который можно получить доступ к параметрам из диалогового окна Смарт-теги. Параметры устанавливаются на уровне книги
`Application.SmartTag-Recognizers`	Коллекция всех установленных классов распознавания соответствует списку классов распознавания в диалоговом окне Смарт-теги
`SmartTagRecognizer`	Установленный класс распознавания, управляющий возможностью его использования
`Worksheet.SmartTags`, `Range.SmartTags`	Коллекция всех смарт-тегов на листе или в диапазоне. Один диапазон может использовать несколько классов распознавания, поэтому содержит несколько смарт-тегов
`SmartTag`	Единственный смарт-тег
`SmartTag.SmartTagActions`	Коллекция операций, допустимых для конкретного смарт-тега. Операции предоставляются классом `Actions`
`SmartTagAction`	Операция смарт-тега в конкретном диапазоне. Для запуска операции используется метод `Execute`
`Workbook.RecheckSmartTags`	Метод для запуска повторного распознавания всех элементов книги

В следующих примерах показано, как использовать некоторые из объектов и свойств.

Проверка активности класса распознавания

При работе со смарт-тегами сначала необходимо убедиться в наличии определенного класса распознавания и проверить, включен ли этот класс.

```
Function IsRecognizerActive(sProgID As String) As Boolean

   Dim oSTR As SmartTagRecognizer

   ' Перебрать все установленные классы распознавания
   For Each oSTR In Application.SmartTagRecognizers

      ' Не этот ли класс мы ищем?
      If oSTR.ProgID = sProgID Then

         ' Если этот, то определить, включен ли он
         IsRecognizerActive = oSTR.Enabled
         Exit For
      End If
   Next

End Function
```

Удаление смарт-тега из диапазона

Одной из проблем работы со смарт-тегами является возможность ложного срабатывания процедуры распознавания, в результате которого в ячейку будет добавлен ошибочный объект SmartTag. Например, стандартный смарт-тег Stock Symbol распознает строку "TRUE" как действительный биржевой символ, даже если это значение True типа Boolean. Следующий код находит и удаляет все результаты ложных срабатываний:

```
Sub RemoveBooleanTrue()

   Dim oSht As Worksheet
   Dim oTag As SmartTag

   ' Это идентификатор URI смарт-тега StockTicker
   Const sTicker As String = _
        "urn:schemas-microsoft-com:office:smarttags#stockticker"

   ' Перебрать все листы активной книги
   For Each oSht In ActiveWorkbook.Worksheets

      'Перебрать все смарт-теги на листе
      For Each oTag In oSht.SmartTags

         ' Смарт-тег StockTicker со значением типа Boolean?
         If oTag.Name = sTicker And _
                       TypeName(oTag.Range.Value) = "Boolean" Then

         ' Да, поэтому смарт-тег необходимо удалить из ячейки
            oTag.Delete
         End If
      Next
   Next

End Sub
```

Добавление смарт-тега в диапазон

При создании класса Recognizer для выполнения распознавания класс имеет глобальную область видимости. Это значит, что метод Recognize будет вызываться для каждого значения во всех книгах Excel или во всех документах Word. Существует возможность ограничить область видимости до Word или Excel, но ограничить область видимости до книги, листа или диапазона нельзя.

Коллекция SmartTags предоставляет метод Add, позволяющий добавлять в ячейку смарт-тег для определенного типа данных. Этот метод дает возможность полностью обойти класс Recognizer и выполнить распознавание с помощью кода на VBA. В таком случае придется реализовать только класс Actions, который предоставляет всплывающее меню и обеспечивает выполнение операций.

Например, может оказаться, что смарт-тег StockTicker распознает слишком много ячеек, на самом деле не содержащих биржевых символов. Предположим, что имеется сводная таблица с полем биржевых символов и только элементы из этого поля должны содержать смарт-теги Stock Symbol. Сначала необходимо отключить класс Recognizer для смарт-тега Stock Symbol (это можно сделать в диалоговом окне Смарт-теги). После этого можно использовать следующий код VBA для распознавания биржевых символов в сводной таблице:

```
Sub AddTickerTagToPivotTableTitles()

   Dim oField As PivotField
   Dim oItem As PivotItem
   Dim oCell As Range

   Const sTicker As String = _
        "urn:schemas-microsoft-com:office:smarttags#stockticker"

   ' Получить 'Символ' из поля сводной таблицы
Set oField = ActiveSheet.PivotTables(1).PivotFields("Symbol")

   ' Перебрать все элементы этого поля сводной таблицы
   For Each oItem In oField.PivotItems

      ' Перебрать все ячейки в диапазоне этого поля сводной таблицы
      For Each oCell In oItem.LabelRange

         ' Если ячейка не пуста
         If Not IsEmpty(oCell) Then

            '... добавить в ячейку смарт-тег Stock Symbol
            oCell.SmartTags.Add sTicker
         End If
      Next
   Next

End Sub
```

Подобный прием позволяет комбинировать интерфейс и мощь всплывающего меню операций объектов SmartTag с точностью и контролем распознавания с помощью Excel VBA. После распознавания объекты SmartTag могут сохраняться в пределах книги. Для этого в диалоговом окне Параметры смарт-тега (SmartTag Options) необходимо установить флажок Внедрить смарт-теги в книгу (Embed SmartTags in this Workbook).

Проблемы, связанные с использованием смарт-тегов

Компания Microsoft реализовала отличную технологию смарт-тегов, но данная реализация имеет несколько слабых мест, о которых стоит упомянуть.

Метод Recognize

Самой значительной проблемой является относительно небольшой объем информации, которая передается в метод Recognize из интерфейса ISmartTagRecognizer.

```
Private Sub ISmartTagRecognizer_Recognize(ByVal Text As String, _
               ByVal DataType As SmartTagLib.IF_TYPE, _
               ByVal LocaleID As Long, _
               ByVal RecognizerSite As
↳ SmartTagLib.ISmartTagRecognizerSite)
```

Обратите внимание, что текст передается в виде строкового значения (As String). При этом отсутствует информация об источнике этого текста. Например, значение True типа Boolean и текстовое значение TRUE могут быть сгенерированы на основе строки "TRUE" — метод Recognizer не различает эти значения. Хотя во второй версии библиотеки типов SmartTag предоставляется больший объем информации и выполняется разбор текста на элементы, информация об источнике текста не предоставляется.

Вторым слабым местом текущей реализации является отсутствие информационного контекста. Библиотека просто получает запрос на распознавание единственного текстового элемента. Если бы была возможность проверить соседние ячейки, распознавание было бы более точным. Например, смарт-тег "Car Model" распознает текст "206" как номер модели Peugeot 206. Если бы была возможность рассмотреть соседние ячейки, можно было бы заметить, что "206" является одним из многих чисел и не имеет никакого отношения к номеру модели.

Третье слабое место связано с тем, что объекту не передается информация о "важности" или позиционном контексте. Например, нет возможности отличать заголовок списка (который не должен содержать смарт-тег) от элемента списка (который может содержать смарт-тег). Точно так же для создания смарт-тегов в конкретной книге (например, созданной на основе конкретного шаблона) или в конкретных ячейках сводной таблицы необходимо использовать код VBA.

Все эти проблемы могут быть решены через передачу того же параметра `Target`, который доступен в методе `InvokeVerb` интерфейса `ISmartTagAction`. Если бы в метод передавалась ячейка с распознаваемым текстом, метод мог бы рассмотреть соседние ячейки, проверить позиционный контекст или ограничить распознавание до определенных книг.

Покрытие

В Office 2003 классу `Recognizer` передается только содержимое ячейки на листе Excel или целый абзац Word. В большинстве приложений Excel используется намного больше представлений текста:

- диаграммы;
- диалоговые окна `UserForm`;
- комментарии, метки и другие элементы управления;
- диалоговые окна VBA `MsgBox` и VBA `InputBox`.

Было бы неплохо, если бы распознавание текста распространялось и на эти формы представления.

Резюме

В Office 2003 компания Microsoft предоставила возможность реализации собственных подпрограмм распознавания введенных данных и собственных меню со списком операций, которые выполняются для распознанных ячеек. Эти операции могут использоваться для обработки распознанных ячеек, поиска необходимой информации из других источников и для простой интеграции данных с листов с другими приложениями.

Классы `Recognizer` и `Actions` легко реализовать любому разработчику, который знаком с языком Visual Basic.

Для точного управления смарт-тегами можно использовать язык VBA. Иногда это позволяет полностью отказаться от применения класса `Recognizer`.

Как всегда при добавлении новой функциональности, в следующей версии остается пространство для улучшений.

Существует несколько сайтов, посвященных использованию смарт-тегов:

- `http://www.officesmarttags.com;`
- `http://www.officezealot.com/smarttags.s`

28

Excel и сеть Internet

Так исторически сложилось, что большинство основанных на Excel приложений оставались внутри самой программы Excel, а внешнее взаимодействие осуществлялось через пользователя, который предоставлял данные и получал результаты. Для хранения данных применялись отдельные книги, которые, как могли, имитировали реляционную базу данных.

В то время как технологии работы с данными развивались от драйверов ODBC в сторону DAO и до текущей версии ADO (дополнительная информация о библиотеке ADO приводится в главе 11), практика хранения данных во внешних базах данных и даже получения данных от других систем по сети стала более распространенной. Теперь Excel все чаще используется в качестве интерфейса запросов и анализа для больших корпоративных баз данных. При этом для получения данных применяются объекты `QueryTable` и `PivotTable`. Но при таком подходе доступность данных ограничивается корпоративной сетью и базами данных.

Начиная с Office 97 компания Microsoft увеличивала способность Excel получать данные из внешних источников, включая доступ к сети Internet и использование смежных технологий. Это была встроенная функциональность (например, Web-запросы) или возможность использования стандартных внешних объектов (например, Internet Transfer Control, управление Web-обозревателем и программа разбора MSXML). Все эти объекты включены в состав пакета Office.

В Excel 2002 предоставлен достаточный объем функциональности для пересмотра подхода к разработке приложений Excel. В результате можно выходить за пределы привычной среды Excel/ADO при получении данных, публикации результатов, контроле за приложениями и совместном использовании данных с распределенными системами за пределами локальной сети.

В этой главе рассматривается функциональность, доступная в Excel 2002. Кроме этого, показано, как использовать часть этой функциональности для доступа к сети Internet. Полное описание связанной с сетью Internet функциональности Excel выходит за пределы тематики этой книги.

> Обратите внимание, что в этой главе термин "Internet" употребляется в самом широком смысле, описывая как внутренние, так и внешние сети. Для чтения этой главы необходимо иметь базовое представление о сети Internet и принципах ее работы. В приведенных примерах используется Web-сервер, работающий на локальном компьютере, но эти приложения могут применяться и для работы с удаленным сервером.

Что за ажиотаж?

Комбинирование Excel и сети Internet позволяет:

- совместно использовать информацию;
- публиковать информацию для неизвестных потребителей. Использование стандартных форматов и протоколов позволяет потребителям получать однородный, надежный и безопасный доступ к информации;
- глобально распространять информацию как внутри, так и за пределами сетей организации, сохраняя управление безопасностью и доступом к ценной информации;
- извлекать информацию, предоставляемую другими организациями или частными лицами. При этом информация может извлекаться из нескольких несвязанных источников и использоваться в качестве исходных данных для приложения;
- совместно использовать информацию производителями и потребителями или поставщиками и клиентами через применение стандартных форматов обмена;
- увеличивать полезность приложение через распространение результатов работы среди большего числа получателей, чем при работе на отдельном персональном компьютере;
- использовать Excel в качестве ключевого компонента бизнес-процессов большего масштаба, включающих в себя несколько организаций.

Использование сети Internet для хранения книг

Самым простым способом совместного использования информации является хранение книг на Web-сервере. Excel 97 предоставляет возможность только загружать книги с Web-сайтов, а в Excel 2000 и Excel 2002 поддерживается также сохранение книг на Web-сервере. Для этого используются расширения FrontPage Server, которые должны работать на сервере. Для открытия книги с Web-сервера или сохранения книги на сервере вместо имени файла используется адрес URL.

```
Sub OpenFromWebSiteAndSaveBack()

    Dim oBk As Workbook

    ' Открыть книгу с Web-сайта
    ' Следующую ссылку необходимо заменить на адрес сервера,
    ' к которому предоставлен доступ
    Set oBk = Workbooks.Open("http://www.MySite.com/book1.xls")

    ' Сохранить книгу на Web-сайте с новым именем
    oBk.SaveAs "http://www.MySite.com/Book2.xls"

End Sub
```

Если сервер требует регистрации, Excel может запрашивать идентификатор и пароль при каждом сеансе доступа (как в предыдущем примере) или передавать идентификатор и пароль в качестве одного из компонентов URL:

```
Sub OpenFromSecureWebSiteAndSaveBack()

    Dim oBk As Workbook

    ' Открыть книгу с Web-сайта
    Set oBk =
↳Workbooks.Open("http://UserID:Pwd@www.MySite.com/book1.xls")

    ' Сохранить книгу на Web-сайте с новым именем
    oBk.SaveAs "http://UserID:Pwd@www.MySite.com/Book2.xls"

End Sub
```

Конечно, адреса URL можно использовать и в диалоговых окнах Открыть файл (File Open) и Сохранить как (Save As).

Использование сети Internet в качестве источника данных

“Классическое” приложение Excel использует два источника данных — базы данных в сети и пользователя. Если элемент данных не доступен в базе данных, пользователь должен ввести и поддерживать этот элемент. Для этого приложение должно содержать листы и диалоговые окна для хранения информации и предоставления механизма для ввода данных.

В качестве типичного примера можно привести поддержку информации о курсе обмена в финансовой модели; обычно ответственность за получение и ввод в модель последних значений курса обмена возлагается на пользователя. Приложение можно сделать более полезным, если автоматизировать получение информации о курсе обмена из одного из Web-сайтов.

В следующих разделах представлены различные способы получения информации из сети Internet. В качестве примера используются значения курса обмена долларов США, доступные на сайте `http://www.x-rates.com/d/USD/table.html`. Ниже показан внешний вид соответствующей Web-страницы (рис. 28.1).

Открытие Web-страниц в качестве книг

Самым простым решением является открытие всей Web-страницы в виде книги с последующим сканированием листов в поисках необходимой информации, например курса обмена USD/GBP.

```
Sub OpenUSDRatesPage()

    Dim oBk As Workbook
    Dim oRng As Range

    ' Открыть страницу в виде книги
    Set oBk = Workbooks.Open("http://www.x-
↳rates.com/d/USD/table.html")

    ' Найти информацию о британском фунте
```

```
    Set oRng = oBk.Worksheets(1).Cells.Find("British Pounds")

    ' Считать значение курса обмена
    MsgBox "Курс обмена USD/GBP составляет " & _
           oRng.Offset(0, 1).Value
End Sub
```

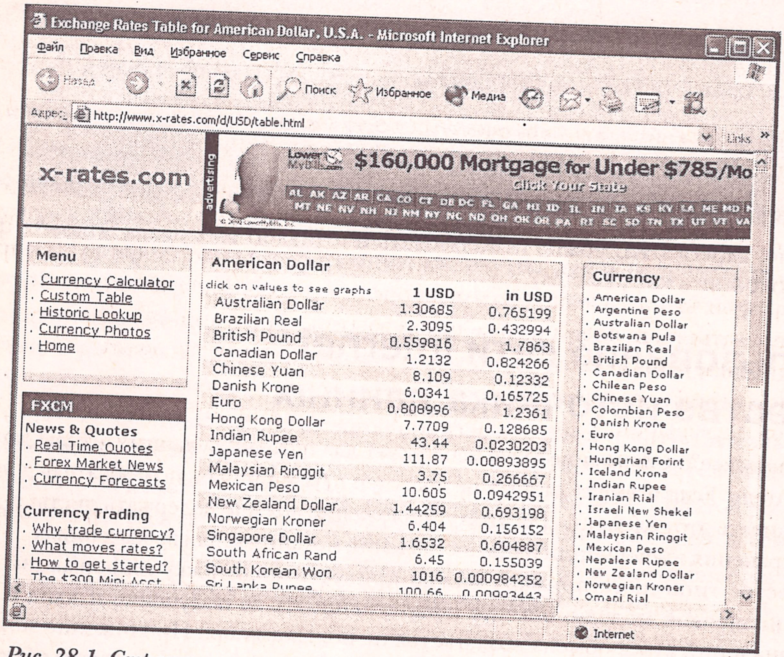


Рис. 28.1. Страница со значениями курса обмена валют

С таким подходом связана определенная проблема – приходится загружать всю Web-страницу (включая графические изображения, рекламные баннеры и т.д.). При этом приходится загружать даже ту информацию, которая не представляет интерес. Ненужные данные значительно замедляют получение необходимой информации.

Использование Web-запросов

Web-запросы впервые появились в Excel 97 и улучшались в каждой последующей версии Excel. Web-запросы позволяют получать единственную таблицу с информацией с Web-страницы. При этом поддерживается возможность автоматического обновления данных при каждом открытии книги или через регулярные промежутки времени.

Одной из проблем Web-запросов было использование десятичного разделителя и разделителя тысяч из региональных параметров Windows при интерпретации чисел, полученных с Web-страницы. Если информация о курсе обмена извлечена из европейского сайта, точка воспринимается как разделитель разрядов, а не разделитель целой и дробной части. В результате курс обмена оказывается на несколько порядков больше. Следовательно, до Excel 2002 Web-запросы не могли обеспечить надежного извлечения ин-

формации, в которой используются разделители целой и дробной части и разделители разрядов, отличные от применяемых в США.

В Excel 2002 компания Microsoft добавила в объект Application три свойства, позволяющих временно переопределить параметры распознавания чисел. Эти свойства доступны и в Excel 2003.

- Application.DecimalSeparator — символ, используемый в качестве разделителя целой и дробной части.
- Application.ThousandsSeparator — символ, используемый в качестве разделителя разрядов.
- Application.UseSystemSeparators — выбор набора разделителей, используемых при интерпретации чисел (Excel или Windows).

Эти свойства позволяют выбрать в Excel разделители, совпадающие с разделителями на Web-странице, выполнить запрос и вернуть разделители, принятые по умолчанию. Если необходимо использовать возможности автоматического обновления Web-запросов, эти разделители должны устанавливаться в процедуре обработки события BeforeRefresh и восстанавливаться в процедуре обработки события AfterRefresh. Создание таких процедур предполагает применение модулей классов для перехвата событий (более подробно эта тема рассматривается в главе 6).

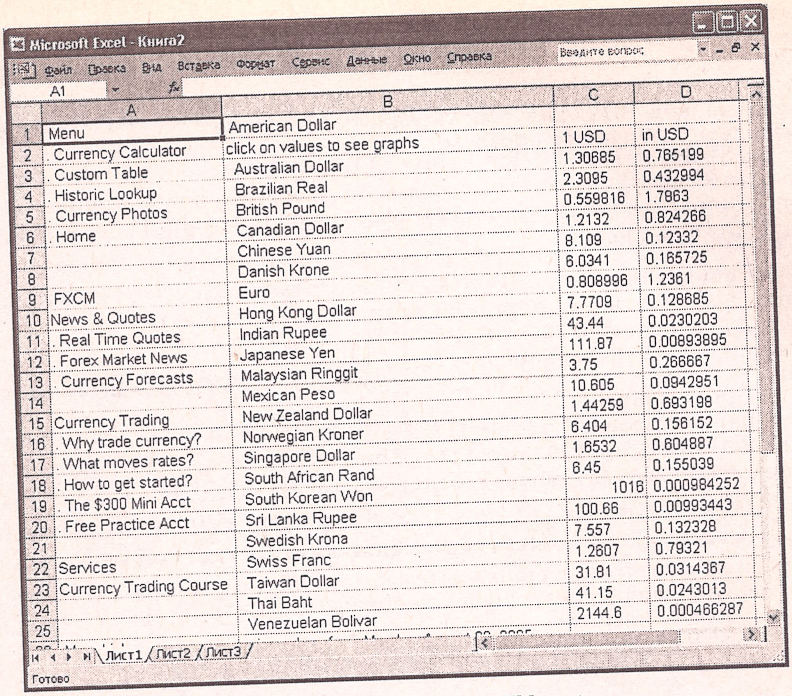
В данном случае достаточно извлечь таблицу с курсами обмена валют. Следующий код создает и выполняет новый Web-запрос. На самом деле проще создать код с помощью механизма записи макроса, что позволит обеспечить правильность выделения.

```
' Извлечение курсов обмена долларов США с помощью Web-запроса
Sub GetRatesWithWebQuery()

    Dim oBk As Workbook
    Dim oQT As QueryTable
    ' Сохранение текущих параметров форматирования чисел
    Dim sDecimal As String
    Dim sThousand As String
    Dim bUseSystem As Boolean

    ' Создание новой книги
    Set oBk = Workbooks.Add

    ' Создание таблицы запроса для загрузки курсов обмена
    ' долларов США
    With oBk.Worksheets(1)
        Set oQT = .QueryTables.Add( _
        Connection:="URL; http://www.x-rates.com/d/USD/table.html", _
        Destination:=.Range("A1"))
    End With

    ' Установка свойств таблицы запроса
    With oQT
        .Name = "USD"

        ' Указать выделение конкретной  таблицы
        .WebSelectionType = xlSpecifiedTables

        ' Импортировать пятую таблицу на странице
        .WebTables = "5"
```

```
        ' Игнорировать форматирование Web-страницы
        .WebFormatting = xlWebFormattingNone

        ' Не пытаться распознавать даты
        .WebDisableDateRecognition = True

        ' Не обновлять запрос при каждом открытии файла
        .RefreshOnFileOpen = False

        ' Ожидать завершения запроса
        .BackgroundQuery = True

        ' Сохранить результаты запроса в книге
        .SaveData = True

        ' Изменить ширину столбца в соответствии с новыми данными
        .AdjustColumnWidth = True
    End With

    With Application
        ' Восстановить текущие параметры форматирования чисел
        sDecimal = .DecimalSeparator
        sThousand = .ThousandsSeparator
        bUseSystem = .UseSystemSeparators

        ' Изменить разделители в соответствии с разделителями
        ' на Web-странице
        .DecimalSeparator = "."
        .ThousandsSeparator = ","
        .UseSystemSeparators = True

        ' Игнорировать ошибки, возникающие в результате
        ' неудачного завершения запроса
        On Error Resume Next

        ' Выполнить запрос, ожидая его завершения
        oQT.Refresh BackgroundQuery:=False
        ' Восстановить параметры форматирования чисел в Excel
        .DecimalSeparator = sDecimal
        .ThousandsSeparator = sThousand
        .UseSystemSeparators = bUseSystem
    End With

End Sub
```

Строка `.WebTables=5` в предыдущем примере заставляет запрос извлекать пятую таблицу на странице. Буквально это означает пятый экземпляр тега `<TABLE>` в исходном коде HTML-страницы (рис. 28.2).

Данный код разбирает Web-страницу и отображает всю необходимую информацию, за исключением графических изображений и побочной информации.



	A	B	C	D
1	Menu	American Dollar	1 USD	in USD
2	Currency Calculator	click on values to see graphs	1.30685	0.765199
3	. Custom Table	Australian Dollar	2.3095	0.432994
4	. Historic Lookup	Brazilian Real	0.559816	1.7863
5	. Currency Photos	British Pound	1.2132	0.824266
6	. Home	Canadian Dollar	8.109	0.12332
7		Chinese Yuan	6.0341	0.165725
8		Danish Krone	0.808996	1.2361
9	FXCM	Euro	7.7709	0.128685
10	News & Quotes	Hong Kong Dollar	43.44	0.0230203
11	. Real Time Quotes	Indian Rupee	111.87	0.00893895
12	. Forex Market News	Japanese Yen	3.75	0.266667
13	. Currency Forecasts	Malaysian Ringgit	10.605	0.0942951
14		Mexican Peso	1.44259	0.693198
15	Currency Trading	New Zealand Dollar	6.404	0.156152
16	. Why trade currency?	Norwegian Kroner	1.6532	0.604887
17	. What moves rates?	Singapore Dollar	6.45	0.155039
18	. How to get started?	South African Rand	1016	0.000984252
19	. The $300 Mini Acct	South Korean Won	100.66	0.00993443
20	. Free Practice Acct	Sri Lanka Rupee	7.557	0.132328
21		Swedish Krona	1.2607	0.79321
22	Services	Swiss Franc	31.81	0.0314367
23	Currency Trading Course	Taiwan Dollar	41.15	0.0243013
24		Thai Baht	2144.6	0.000466287
25		Venezuelan Bolivar		

Рис. 28.2. Таблица с результатом выполнения Web-запроса

Разбор Web-страницы для получения конкретной информации

Web-запросы отлично справляются с получением таблиц с информацией, находящихся на Web-страницах. Но Web-запросы слишком громоздки, если необходимо получить один или два элемента массива информации. Еще одним решением является считывание страницы с помощью скрытого экземпляра Internet Explorer, поиск необходимой информации в исходном коде страницы и возврат интересующего результата. Следующий код требует наличия ссылки на объектную библиотеку “Microsoft Internet Controls”.

```
Sub GetUSDtoGBPRateUsingIE()

    Dim oIE As SHDocVw.InternetExplorer
    Dim sPage As String
    Dim iGBP As Long, iDec As Long

    Dim iStart As Long, iEnd As Long
    Dim dRate As Double

    ' Создать новый (скрытый) экземпляр IE
    Set oIE = New SHDocVw.InternetExplorer

    ' Открыть Web-страницу
    oIE.Navigate "http://www.x-rates.com/d/USD/table.html"

    ' Подождать полной загрузки Web-страницы
    Do Until oIE.readyState = READYSTATE_COMPLETE
        DoEvents
    Loop
```

```
    ' Извлечь текст Web-страницы в переменную
    sPage = oIE.Document.body.InnerHTML

'***********************************************************
    ' Исходный код HTML для британских фунтов будет
    ' выглядеть следующим образом:
    '<tr bgcolor=white>
    '<td>
    '<font face="Verdana" size=-1>  British
↳ Pound  </font>
    '</td>
    '<td align="right">
    '<font face="Verdana" size=-1> <a
↳href="/d/GBP/USD/graph120.html"
↳class="menu">0.54936</a> </font>
    '</td>
    '<td align="right">
    '<font face="Verdana" size=-1> <a
↳href="/d/USD/GBP/graph120.html"
↳class="menu">1.8203</a> </font>
    '</td>
    '</tr>
'***********************************************************

    ' Для определения курса обмена необходимо найти
    ' запись для британских фунтов
    ' после чего переместиться вперед с получения
    ' значения курса обмена

    ' Найти запись для британских фунтов в строке HTML
    iGBP = InStr(1, sPage, " British Pound")

    ' Найти следующий десятичный разделитель, который
    ' находится в расширении файла HTML
    iDec = InStr(iGBP, sPage, ".")
    ' Найти следующий десятичный  разделитель, который
    ' будет находится в пределах значения курса обмена
    iDec = InStr(iDec + 1, sPage, ".")

    ' Найти начало и конец числа
    iStart = InStrRev(sPage, ">", iDec) + 1
    iEnd = InStr(iDec, sPage, "<")

    ' Оценить число, предположив, что оно соответствует
    ' региональным параметрам США
    dRate = Val(Mid$(sPage, iStart, iEnd - iStart))

    ' Вывести курс обмена
    MsgBox "Курс обмена USD/GBP равен " & dRate

End Sub
```

Самый подходящий метод зависит от конкретных обстоятельств, а также от необходимого объема данных. Для единственного элемента желательно использовать последний подход. Для получения большего количества элементов стоит применять Web-запрос для перенесения страницы или таблицы в книгу и поиска интересующих записей на листе.

Использование сети Internet для публикации результатов

Web-сервер может использоваться в качестве хранилища информации. Результаты работы приложения могут храниться на Web-сервере и предоставляться большей аудитории, чем при использовании печатных версий отчетов. Размещение результатов на Web-страницах позволяет читателям применять эти результаты в качестве источников данных для собственного анализа и передачи результатов этого анализа заинтересованным сторонам.

Настройка Web-сервера

Все показанные ниже примеры требуют привилегий записи на Web-сервер. Так как в некоторых примерах используются Active Server Pages (ASP), то в данном случае будет использоваться сервер Microsoft IIS 5.1. Запустите консоль IIS и щелкните правой кнопкой мыши на узле Default Web site. Выберите пункт Properties (Свойства) и активизируйте вкладку Home Directory (Домашний каталог) (рис. 28.3). При этом будут предоставлены различные параметры принятого по умолчанию Web-сайта. Установите флажки Read (Чтение) и Write (Запись) и щелкните на кнопке OK.

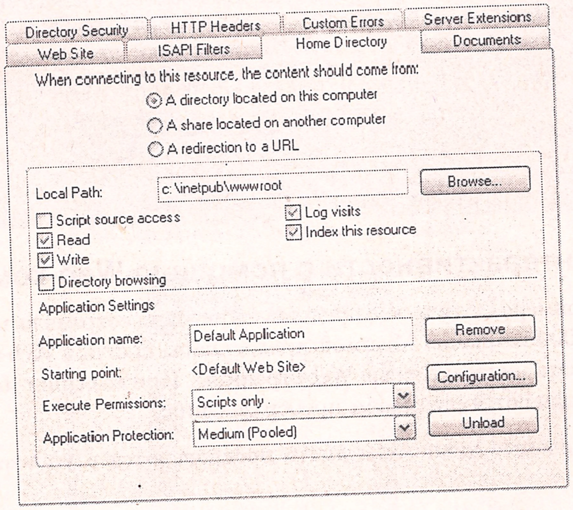


Рис. 28.3. Настройка параметров Web-сайта

> Сервер IIS 5.1 предоставляется в составе Windows XP. Для запуска консоли IIS откройте меню Пуск (Start), выберите пункт Выполнить (Run) и введите команду `Inetmgr`. После этого щелкните на кнопке OK. Вместо этого можно открыть Панель управления (Control Panel), выполнить двойной щелчок на пиктограмме Администрирование (Administrative Tools) и запустить консоль Internet Information Services.

Обратите внимание на поле Local Path: (Локальный путь). Именно в нем определяется корневой каталог Web-сервера. По умолчанию в качестве корневого каталога используется `C:\inetpub\wwwroot\`. Любая Web-страница, расположенная в этом каталоге, доступна

по адресу http://localhost/<Имя страницы>.html. Именно на этой странице будут опубликованы результаты первых нескольких примеров. Пример Timesheet в этой главе потребует настройки виртуального каталога (Virtual Directory), что будет рассмотрено далее.

Сохранение листов в виде Web-страниц

Самым простым способом публикации результатов в виде Web-страницы является создание шаблона книги, в котором определено все форматирование и ссылки. После этого при получении необходимых результатов достаточно скопировать соответствующие значения и сохранить книгу непосредственно на Web-сервере.

```
Sub PublishResultsToWeb()

    Dim oBk As Workbook
    Dim oSht As Worksheet

    ' Создать новую копию книги Web Template
    Set oBk = Workbooks.Add("c:\mydir\WebTemplate.xls")

    ' Получить первый лист книги
    Set oSht = oBk.Worksheets(1)

    ' Внести результаты в книгу-шаблон
    oSht.Range("Profits").Value = Workbooks("Results.xls") _
          .Worksheets("Financials").Range("Profits").Value

    ' Сохранить непосредственно на сервере как Web-страницу
    oSht.SaveAs "http://localhost/ResultsJuly2001.htm", xlHtml

    ' Закрыть книгу
    oBk.Close False

End Sub
```

Добавление интерактивности с помощью Web-компонентов

В предыдущем примере статическое представление листа сохранялось на сервере в формате HTML. В Excel 2000 компания Microsoft предоставила доступ к Web-компонентам Office, позволяющим создавать интерактивные Web-страницы. При создании интерактивного представления листа выполняются следующие преобразования:

- лист или отдельные диапазоны листа превращаются в Web-компонент Spreadsheet;
- выделенные диаграммы превращаются в Web-компоненты Chart;
- сводные таблицы превращаются в Web-компоненты PivotTable.

Эти компоненты являются встроенными в HTML-страницу элементами управления ActiveX, обеспечивающими тот же уровень интерактивности, что и пользовательский интерфейс Excel, но доступными через обозреватель.

Подробное описание Web-компонентов выходит за пределы тематики этой книги (в Office 2003 механизм Web-компонентов был значительно улучшен), но следующий код можно использовать для сохранения книги в виде интерактивной Web-страницы. Такая книга может содержать опубликованный диапазон данных (A1:C30), сводную таблицу и встроенную диаграмму.

```
Sub PublishPageInteractive()

  ' Коллекция  PublishObjects содержит все компоненты листа,
  ' которые должны быть опубликованы на Web-странице.
  With ActiveWorkbook.PublishObjects

    ' Удалить все объекты публикации
    .Delete

   ' Начать с Web-компонента Pivot Table
   .Add(xlSourcePivotTable, "http://localhost/page.htm", "Sheet1", _
        "PivotTable1", xlHtmlList).Publish True

    ' добавить диапазон листа на эту же страницу
    .Add(xlSourceRange, "http://localhost/page.htm", "Sheet1", _
        "A1:C30", xlHtmlCalc).Publish False

    ' После этого публикуется диаграмма с исходными данными
    .Add(xlSourceChart, "http://localhost/page.htm", "Sheet1", _
        "Chart 1", xlHtmlChart).Publish False
  End With

End Sub
```

Обратите внимание, что в предыдущем примере все компоненты публикуются на одной и той же странице. Указание различных адресов URL позволяет создать несколько страниц.

В результате работы кода создаются достаточно простые Web-страницы — они просто используются для размещения различных Web-компонентов (рис. 28.4). Скорее всего, для создания качественных Web-страниц потребуется дополнительная обработка.

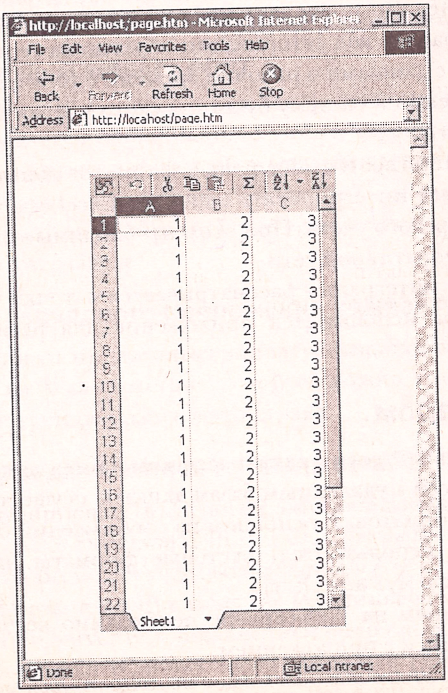


Рис. 28.4. Публикация элементов книги в виде Web-страницы

Использование сети Internet в качестве канала связи

Извлечение данных из Web-страниц и публикация результатов в виде Web-страниц во многих отношениях является пассивным использованием сети Internet; при этом Web-сервер в основном служит в качестве хранилища. Кроме этого, Web-серверы могут поддерживать работу приложений, которые обеспечивают более динамическое взаимодействие с пользователями. Серверное приложение выступает в роли главной контактной точки для всех клиентских книг. При этом сервер выполняет следующие функции:

- централизованное хранение данных;
- сведение данных от нескольких клиентов;
- распространение данных среди остальных клиентов;
- управление процессами;
- поддержка механизмов обработки значений.

В качестве примера можно рассмотреть систему учета рабочего времени, в которой каждому сотруднику выделяется книга Excel для ежедневного ввода информации об отработанном времени. В конце каждого месяца сотрудники подключаются к сети Internet и отправляют книги с информацией о рабочем времени приложению на Web-сервере. При этом серверное приложение сохраняет полученные данные в центральной базе данных. После этого менеджер подключается к серверу и получает информацию о рабочем времени персонала. Менеджер проверяет предоставленную информацию и выдает разрешение на оплату, отправляя код авторизации обратно на сервер. Бухгалтерия извлекает авторизованный лист с данными о рабочем времени с того же Web-сервера и загружает эти данные непосредственно в систему бухгалтерского учета с последующей обработкой выплаты заработной платы.

Использование сети Internet и стандартных форматов данных при двунаправленном обмене данными позволяет интегрировать клиентов Excel с другими системами, например, с системой бухгалтерского учета. При этом необходимые бизнес-процессы могут работать за пределами корпоративной сети.

В этом разделе такая интеграция рассматривается с точки зрения Excel 2003. При этом в качестве примера используется приложение для регистрации сообщений об ошибках.

Связь с Web-сервером

В пределах корпоративной сети практически весь обмен данными, от передачи файлов до выполнения запросов к удаленным базам данных, осуществляется с использованием закрытых двоичных форматов. В основном из соображений безопасности для обмена данными в сети Internet используются текстовые форматы, например адреса URL — `http://www.MySite.com/MyPage.htm`.

Для связи с приложением на Web-сервере необходимо вести обработку на сервере и обмениваться информацией с приложением.

В Excel 2003 для связи с Web-сервером может использоваться метод `FollowHyperlink` объекта `Workbook`. Но применение этого метода связано с рядом проблем, включая:

- все данные, полученные в результате использования гиперссылки, автоматически отображаются в виде новой книги;
- разработчик практически не управляет процессом обмена информацией.

Компонент Microsoft Internet Transfer Control (`msinet.ocx`) является более гибкой альтернативой. Элемент управления ActiveX, который часто называют ITC, является оболочкой над файлом `wininet.dll`, предоставляющим низкоуровневые функции сети Internet на платформе Windows.

Передача данных от клиента к серверному приложению

Существует два механизма, позволяющих отправлять информацию на Web-сервер. Ее можно передавать в составе строки URL или отправлять в качестве отдельного фрагмента запроса HTTP.

Кодирование URL

Параметры могут кодироваться в строке URL. Для этого перед первым параметром после строки URL указывается символ ?. Для разделения параметров между собой используется символ &.

```
http://www.MySite.com/MyPage.asp?param1=value1
↳&param2=value2&param2=value3
```

Преимущество этого метода заключается в том, что параметры остаются частью строки URL и могут сохраняться в папке Избранное (Favorites) или вводиться непосредственно в обозреватель. Недостатком этого способа является ограничение на длину строки URL (в Internet Explorer длина строки URL не может превышать 2083 символа), что сужает объем информации, передаваемой в качестве параметров.

Передача данных с помощью запроса POST

При отправке запроса на Web-сервер в нем содержится информация о различных заголовках, в которых передается информация о типе и версии клиентского приложения, типе и версии протокола связи, а также об идентификаторе пользователя и паролях. Кроме этого, в запросе может присутствовать поле POST, которое желательно использовать для передачи информации на сервер. Применение этого способа показано в примере приложения далее в этом разделе.

Передача данных от серверного приложения к клиенту

Передача информации от приложения клиенту выполняется очень просто — данные могут предоставляться на Web-странице, которую можно считывать с помощью стандартных методов, рассмотренных ранее в этой главе.

Приложение Web-сервера — запись журнала ошибок

Существует ряд конкурирующих технологий, обеспечивающих возможность обработки данных на Web-сервере: .NET Framework, PERL, CGI, JavaServer Pages (JSP) и Active Server Pages (ASP). Разработчик VBA может создавать страницы ASP с помощью языка VBScript.

Для предоставления страниц ASP в качестве Web-сервера должен использоваться Microsoft Internet Information Server под управлением операционной системы Windows NT, Windows 2000, Windows XP или Windows Server 2003, или Personal Web server под управлением операционной системы Windows 98 или Windows Me. Дополнительную информацию можно найти в разделе “Настройка Web-сервера” ранее в этой главе.

В следующем примере показано создание приложения для Web-сервера, обеспечивающего работу центрального журнала для любых ошибок времени выполнения, возникающих в процессе работы приложения Excel (или любого другого приложения). Серверное приложение состоит из трех компонентов:

- базы данных Access, в которой хранится журнал ошибок;
- активной серверной страницы для записи в журнал ошибок;
- активной серверной страницы для отображения содержимого журнала ошибок.

Клиент (Excel) использует общую процедуру обработки ошибок.

База данных Access для хранения журнала ошибок

Сначала создайте базу данных `ErrorLog.mdb` в Access 2002. База данных должна содержать таблицу `ErrorData` со следующими полями:

Имя поля	*Тип данных*	*Комментарий*
`ErrorID`	`AutoNumber`	Сделайте это поле первичным ключом
`ServerTime`	`Date/Time`	
`Application`	`Text`	50 символов
`ErrSource`	`Memo`	
`ErrNumber`	`Number`	Long Integer
`ErrDescription`	`Memo`	

В этом примере в журнал записывается фактическая информация об ошибках. Обычно в реальных приложениях хранится значительно больший объем информации, включая имя пользователя, имена файлов с данными, имена книг и модулей, версии Excel и Windows, а также региональные параметры.

С помощью FrontPage 2002 создайте новую Web-страницу и импортируйте базу данных. Для этого воспользуйтесь командой File⇨Import... (Файл⇨Импорт). FrontPage автоматически создаст папку `fpdb` для хранения базы данных и подключение к базе данных. При этом у разработчика будет запрошено имя для создаваемого подключения. Назовите подключение `ErrorLog`.

Виртуальные каталоги

Как было показано ранее в разделе "Настройка Web-сервера", в данном примере будут использоваться виртуальные каталоги. Откройте IIS и щелкните правой кнопкой мыши на узле Default Web. Выберите пункт New⇨Virtual Directory (Создать⇨Виртуальный каталог). После этого запустится мастер создания виртуального каталога (Virtual Directory Creation Wizard). Щелкните на кнопке Next (Далее) и введите псевдоним **`VBA_ProgRef`** (рис. 28.5).

При этом URL `http://localhost/VBS_ProgRef` будет перенаправляться на любой указанный каталог на жестком диске. Имя этого каталога передается серверу IIS на следующем этапе. Введите имя каталога, в котором FrontPage сохранил страницы ASP (рис. 28.6).

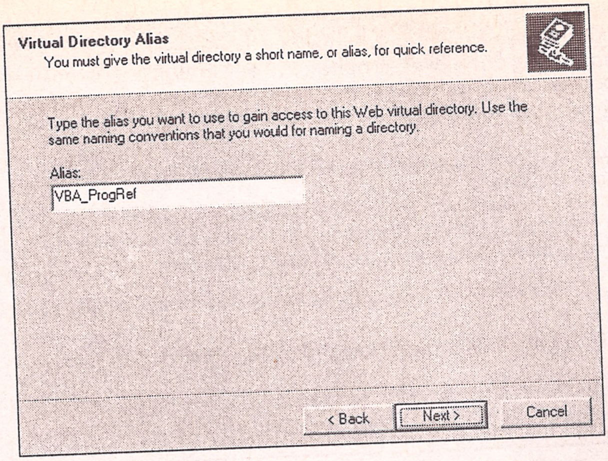


Рис. 28.5. Ввод псевдонима для виртуального каталога

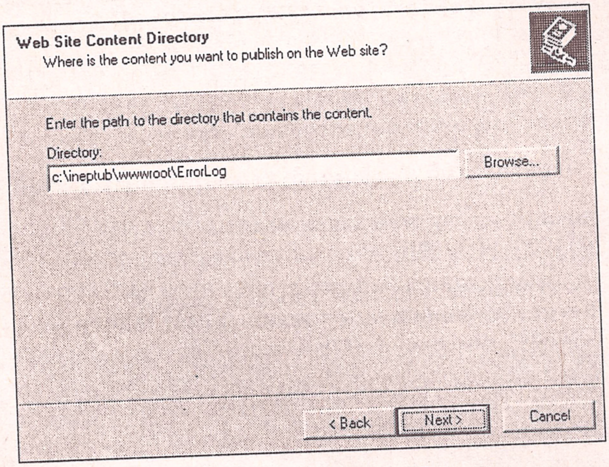


Рис. 28.6. Выбор каталога на жестком диске

Щелкните на кнопке Next (Далее) и проверьте доступность каталога на запись. Это завершающий этап создания виртуального каталога. После этого адрес URL `http://localhost/VBA_ProgRef/MyPage.asp` позволит получать доступ к страницам в каталоге `C:\My Documents\My Webs\myweb\`.

Страница ASP, записывающая данные в журнал ошибок

Создайте пустую страницу во FrontPage. Сохраните страницу под именем `StoreError.asp`, активизируйте вкладку HTML и скопируйте следующий код HTML:

```
<%@ LANGUAGE=VBSCRIPT %>
<%
'*************************************************************
'*
'* Имя страницы: StoreError.asp
'*
```

```
'* Входные параметры:    информация для сохранения
'* в журнале ошибок передается с помощью запроса POST
'* (который используется в Web-формах). Файл POST
'* содержит информацию в формате:
'*
'*           Имя1=Значение1&Имя2=Значение2&Имя3=Значение3 и т.д.
'* Выходные значения:   Возвращает номер созданной записи с
'* информацией об ошибке
'*
'* Назначение:   Получает информацию об ошибках приложения и
'* сохраняет полученную информацию в базе данных
'*
'*************************************************************

    Dim objConn, objRS, sTable, lErrID, objField, objID

    ' Игнорировать любые ошибки, например, отсутствие некоторых
    ' полей в запросе POST
    On Error Resume Next

    ' Создать ссылку на объекты Jet Engine для очистки кэшей
    Set objJE = Server.CreateObject("JRO.JetEngine")

    ' Создать подключение ADO
    Set objConn = Server.CreateObject("ADODB.Connection")

    ' Воспользоваться строкой подключения, сгенерированной
    ' во FrontPage
    objConn.Open Application("ErrorLog_ConnectionString")

    ' Очистить кэш для принудительной записи
    objJE.RefreshCache objConn

    ' Создать подключение к таблице журнала ошибок
    Set objRS = Server.CreateObject("ADODB.Recordset")
    objRS.ActiveConnection = objConn
    objRS.LockType = 3
    objRS.CursorType = 2
    objRS.Source = "ErrorData"
    objRS.Open

    ' Добавить запись в таблицу журнала ошибок
    objRS.AddNew

    ' Внести всю информацию об ошибке
    For Each objField in objRS.Fields

        Select Case objField.Type
        Case 3    ' Long Integer
            If LCase(objField.Name) <> "errorid" Then
                objField.Value = CLng(Request(objField.Name))
            End If
        Case 7, 135    'Date
            If LCase(objField.Name) = "servertime" Then
                objField.Value = Now()
            Else
                objField.Value = CDate(Request(objField.Name))
            End If
        Case 202, 203 'Char, Memo
            objField.Value = Request(objField.Name)
```

```
        End Select
    Next

    ' Сохранить данные в базе данных
    objRS.Update

    ' Считать идентификатор ошибки и вернуть идентификатор клиенту
objRS.MoveLast
Response.Write objRS.Fields(0).Value
objRS.close
   Set objRS = Nothing

   ' Очистить кэш для принудительной записи
   objJE.RefreshCache objConn

   objConn.Close
   Set objConn = Nothing

   Set objJE = Nothing
%>
```

Код просматривает файл базы данных на сервере и ожидает, что для каждого поля в базе данных в запросе POST будет присутствовать параметр. Функция `Request` извлекает значение параметра из строки POST. Извлеченное значение записывается в базу данных. В завершение работы подпрограммы вызов `Response.Write` отправляет идентификатор клиенту.

Страница ASP для просмотра журнала ошибок

Сценарий страницы ASP для просмотра журнала ошибок достаточно сложен, так как он генерирует код HTML для просмотра страницы в обозревателе, а также предоставляет возможность перемещения между ошибками. Создайте новую страницу `ErrorLog.asp`, переключитесь в представление HTML и скопируйте следующий код:

```
<%@ LANGUAGE=VBSCRIPT %>
<%

'**********************************************************
'*
'* Имя страницы: ErrorLog.asp
'*
'* Входные параметры:    ErrID - номер ошибки, которая должна
'* быть показана в обозревателе
'*                       Op    - операция по перемещению
'*
'* Выходные значения:   Выводит информацию об ошибке в окне
'* обозревателя
'*
'* Назначение:   Обеспечивает возможность просмотра базы
'* данных журнала ошибок
'*
'**********************************************************

    ' Любые ошибки будут игнорироваться
    On Error Resume Next

    ' Считать параметры, переданные в URL

    sOp = Request.QueryString("Op")
    iErr = Request.QueryString("ErrID")
```

```
    ' Проверить операцию навигации
    If sOp = "" Then          If iErr = "" Then
            sOp = "Last"
        Else
            sOp = "Go"
        End If
    End If

    If iErr = "" Then iErr = 1
    If iErr < 1 Then iErr = 1

    ' Создать подключение к базе данных журнала ошибок
    Set objConn = Server.CreateObject("ADODB.Connection")
    objConn.Open Application("ErrorLog_ConnectionString")

    ' Создать подключение к таблице журнала ошибок
    Set objRS = Server.CreateObject("ADODB.Recordset")
    objRS.ActiveConnection = objConn
    objRS.CursorType = 3                    ' Статический курсор

    objRS.Source = "ErrorData"
    objRS.Open

    ' Выбрать ошибки, которые будут показаны
    If objRS.BOF And objRS.EOF Then
        iErr = 0
    Else

        ' Найти необходимую ошибку
        objRS.Find "ErrorID = " & iErr

        vBM = objRS.Bookmark

        ' Перейти к последней, следующей или предыдущей
        ' ошибке в зависимости от запрошенной операции
        If objRS.EOF Or sOp = "Last" Then
            objRS.MoveLast

        ElseIf sOp = "Next" Then
            objRS.MoveNext

        ElseIf sOp = "Prev" Then
            objRS.MovePrevious
        End If

        If objRS.EOF Or objRS.BOF Then objRS.Bookmark = vBM

        ' Запомнить отображаемый номер ошибки
        iErr = objRS.Fields("ErrorID")
    End If
%>
<%' Записать информацию о заголовке HTML для Web-страницы %>
<HTML>

<head>
<title>Обозреватель журнала ошибок для Excel 2003</title>
</head>

<%' Создать Web-форму, чтобы пользователь мог перейти
    к конкретной ошибке по идентификатору %>
<FORM METHOD="GET" ACTION="ErrorLog.asp">
```

```
</form>

<table border="0" width="100%">
  <tr>
  <td colspan="2"><p align="center"><b>
    <font face="Arial" size="6">Excel 2003 Prog Ref</font>
    <br>
    <font face="Arial" size="4">Обозреватель журнала
↳ ошибок</font></b>
</td>
  </tr>
  <tr>
    <%' Заполнить поле "Error ID" и ссылки Перейти,
↳ Предыдущая и Следующая %>
    <td width="20%">Номер ошибки: </td>
    <td width="80%"><INPUT TYPE=TEXT NAME="ErrID"
↳ VALUE="<%=iErr%>" SIZE=10>
       <INPUT TYPE=SUBMIT VALUE="Go">
      <a href="ErrorLog.asp?App=<%=sApp%>&ErrID=<%=iErr%>&Op=Prev">
↳Previous</a>-<a href="ErrorLog.asp?App=<%=sApp%>
↳&ErrID=<%=iErr%>&Op=Next">Next</a>
 - <a href="ErrorLog.asp?App=<%=sApp%>&Op=Last">Last</a></td>
  </tr>
<%
    ' Перебрать все поля в таблице, отображая имя и
    ' значение на странице
    For iField = 1 To objRS.Fields.Count - 1
%>
  <tr>
    <td width="20%"><% =objRS.Fields(iField).Name %>: </td>
    <td width="80%">
<%      If Not objRS.EOF Then
%>
        <%=objRS.Fields(iField).Value %>
<%
    End if
%>  </td>
  </tr>
<%
    Next

    ' Очистка объекта ADO
    objRS.Close
    Set objRS = Nothing

    objConn.Close
    Set objConn = Nothing

%>
</FORM>
</table>

</body>
</HTML>
```

Подпрограмма обработки ошибок в Excel

В то время как база данных и две страницы ASP работают на сервере, необходимо написать подпрограмму обработки ошибок в Excel для регистрации ошибок времени выполнения в базе данных на сервере. Для отправки информации об ошибках на приложение Web-сервера используется объект Microsoft Internet Transfer Control. Кроме этого,

ошибка отображается в простом окне сообщения. В реальных приложениях на сервер отправляется больший объем информации, а пользователю выдается более понятное сообщение об ошибке.

Откройте книгу в Excel 2003 и добавьте ссылку на Microsoft Internet Transfer Control (возможно, для этого придется найти файл `msinet.ocx`, который обычно расположен в каталоге `Windows\System32`). Скопируйте в модуль следующий код:

```
Option Explicit

' Общий обработчик ошибок
Sub ErrorHandler()

    Dim oInet As Inet
    Dim lContent As Long
    Dim sData As String
    Dim sHeader As String
    Dim sResult As String

    ' Создать новый экземпляр объекта Internet Transfer Control
    Set oInet = New Inet

    ' Создать строку запроса POST с информацией об ошибке
    sData = "Application=Excel 2003 Prog Ref" & "&" & _
            "ErrSource=" & Err.Source & "&" & _
            "ErrNumber=" & Err.Number & "&" & _
            "ErrDescription=" & Err.Description

    ' Пробелы необходимо заменить на символ +
    sData = Replace(sData, " ", "+")

    ' Указать передачу в запросе POST закодированного
    ' списка параметров
    sHeader = "Content-Type: application/x-www-form-urlencoded"

    ' Отправить информацию об ошибке на сервер
    oInet.AccessType = icDirect
    oInet.Execute "http://localhost/VBA_ProgRef/StoreError.asp", "POST", _
                  sData, sHeader

    ' Подождать, пока сервер не завершит обработку
    Do While oInet.StillExecuting
        DoEvents
    Loop

    ' Извлечь полученный текст (например, идентификатор ошибки)
    lContent = oInet.GetHeader("content-length")
    sResult = oInet.GetChunk(lContent + 100)

    ' Показать идентификатор ошибки
    MsgBox "В базе данных сохранена ошибка номер " & sResult

End Sub

' Намеренно создать ошибку деления на ноль для проверки
' работоспособности обработчика ошибок
Sub TestIt()

    On Error GoTo errHandler

    ' Принудительное деление на ноль
```

```
    Debug.Print 1 / 0

    Exit Sub

errHandler:

    ' Вызвать общую процедуру обработки ошибок
    ErrorHandler

End Sub
```

Запустите подпрограмму `TestIt` несколько раз для проверки процедуры обработки ошибки. Если все настроено правильно, будут выдаваться сообщения с увеличивающимися идентификаторами ошибок. Передача информации между клиентом и сервером осуществляется с помощью списка пар параметр/значение. Компоненты слабо связаны, поэтому изменения в одном компоненте не потребуют изменений других компонентов. Такая слабая связанность и является основным преимуществом Web-приложений.

XML

С усложнением систем между ними стали передаваться все большие объемы информации. На определенном этапе списков параметр/значение становится недостаточно для передачи всего объема информации. Для сохранения слабой связанности Web-архитектуры был разработан новый формат для обмена данными. Этот формат называется eXtensible Markup Language или XML.

Формат XML позволяет структурировать данные в иерархические отношения и назначать каждому элементу определенный тип данных (например, Name, HoursWorked и т.д.). Код XML для простой карточки учета рабочего времени может выглядеть следующим образом (при этом префикс `TS:` используется для отделения собственных тегов от других тегов).

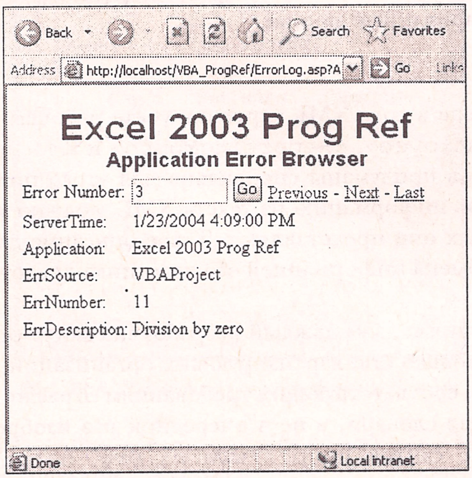


Рис. 28.7. Просмотр содержимого базы данных с информацией об ошибках

```
<?xml version="1.0"?>
<!Excel 2003 Prog Ref Timesheet>
<TS:Timesheet xmlns:TS="urn:schemas_wrox_com:ExcelProgRef:Timesheet">
    <TS:Employee>
        <TS:FirstName>Stephen</TS:FirstName>
        <TS:LastName>Bullen</TS:LastName>
        <TS:EmployeeNumber>1234</TS:EmployeeNumber>
    </TS:Employee>
    <TS:ReportingMonth>
        <TS:Year>2001</TS:Year>
        <TS:Month>6</TS:Month>
        <TS:TotalStdHours>157.5</TS:TotalStdHours>
    </TS:ReportingMonth>
    <TS:HoursWorked>
        <TS:Day>1</TS:Day>
        <TS:StdHours>7.5</TS:StdHours>
        <TS:Overtime>2</TS:Overtime>
    </TS:HoursWorked>
    <TS:HoursWorked>
        <TS:Day>4</TS:Day>
        <TS:StdHours>7.5</TS:StdHours>
        <TS:Overtime>1</TS:Overtime>
    </TS:HoursWorked>
    <TS:HoursWorked>
        <TS:Day>5</TS:Day>
        <TS:StdHours>6</TS:StdHours>
        <TS:Overtime/>
    </TS:HoursWorked>
</TS:Timesheet>
```

Для отправки данных приложению Web-сервера можно воспользоваться все тем же механизмом запросов POST, который применялся для отправки пар параметр/значение. Тег `<?xml version="1.0"?>` в начале кода идентифицирует код как XML. При этом приложение на сервере загрузит код XML в программу разбора (XML Parser), позволяющую извлекать структурированные данные.

Словари XML

В предыдущем примере в коде XML карточки учета рабочего времени используются теги <Timesheet>, <Employee>, <ReportingMonth> и т.д. с произвольно выбранными именами. Имена тегов придуманы специально для этого примера. Любое приложение, пытающееся извлечь информацию из текста XML, должно знать, что означают эти теги и какие типы данных они представляют. Такое описание данных XML называется "Словарем XML". Для обмена информацией оба участника диалога должны использовать один и тот же словарь.

Хотя любая организация и даже каждый разработчик могут определить собственный словарь, существует несколько стандартизирующих организаций, работающих над определением словарей XML, соответствующих требованиям отрасли. По возможности стоит использовать стандартные словари, а не в очередной раз изобретать велосипед. Начинайте поиск словарей с сайтов http://www.w3.org, http://www.BizTalk.org, http://www.rosettanet.org или с сайтов регулирующих органов конкретной отрасли.

> Дополнительная информация об XML доступна в книге *Professional XML, Second Edition* от издательства Wrox Press, ISBN 1-861005-05-9.

Схема XML-SS

Одним из таких словарей XML является схема XML-SS, которая используется компанией Microsoft для описания данных из электронных таблиц. И Excel 2003, и библиотека Office 11 Speadsheet Web Component могут читать и записывать XML с помощью схемы XML-SS. До последнего момента документация на схему XML-SS была доступна на Web-сайте Microsoft. Но в результате одной из реорганизаций эта документация исчезла.

Для получения представления объекта Excel Range в соответствии со схемой XML-SS воспользуйтесь новым аргументом свойства Range.Value:

```
Sub GetRangeXML()

   Dim sRangeXML As String

   sRangeXML = ActiveSheet.Range("A1:C40") _
               .Value(xlRangeValueXMLSpreadsheet)

   ' Печать в окне Immediate
   Debug.Print sRangeXML

End Sub
```

Рассмотрим следующую карточку учета рабочего времени в качестве примера (рис. 28.8).

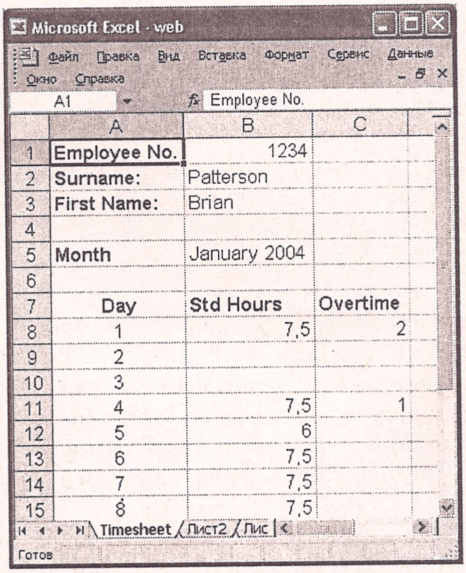


Рис. 28.8. Карточка учета рабочего времени

В результате преобразования в соответствии со схемой XML-SS будет получен следующий код XML (некоторые дни удалены).

```
<?xml version="1.0"?>
<?mso-application progid="Excel.Sheet"?>
<Workbook xmlns="urn:schemas-microsoft-com:office:spreadsheet"
 xmlns:o="urn:schemas-microsoft-com:office:office"
 xmlns:x="urn:schemas-microsoft-com:office:excel"
 xmlns:ss="urn:schemas-microsoft-com:office:spreadsheet"
```

```
  xmlns:html="http://www.w3.org/TR/REC-html40">
  <DocumentProperties xmlns="urn:schemas-microsoft-com:office:office">
   <Author>Brian Patterson</Author>
   <LastAuthor>Brian Patterson</LastAuthor>
   <Created>2004-01-25T22:20:36Z</Created>
   <LastSaved>2004-01-25T22:23:53Z</LastSaved>
   <Version>11.5606</Version>
  </DocumentProperties>
  <ExcelWorkbook xmlns="urn:schemas-microsoft-com:office:excel">
   <WindowHeight>6810</WindowHeight>
   <WindowWidth>4635</WindowWidth>
   <WindowTopX>120</WindowTopX>
   <WindowTopY>15</WindowTopY>
   <ProtectStructure>False</ProtectStructure>
   <ProtectWindows>False</ProtectWindows>
  </ExcelWorkbook>
  <Styles>
   <Style ss:ID="Default" ss:Name="Normal">
   <Alignment ss:Vertical="Bottom"/>
   <Borders/>
   <Font/>
   <Interior/>
   <NumberFormat/>
   <Protection/>
  </Style>
  <Style ss:ID="s22">
   <Font x:Family="Swiss" ss:Bold="1"/>
  </Style>
  <Style ss:ID="s23">
   <Alignment ss:Horizontal="Center" ss:Vertical="Bottom"/>
   <Font x:Family="Swiss" ss:Bold="1"/>
  </Style>
  <Style ss:ID="s24">
   <Alignment ss:Horizontal="Center" ss:Vertical="Bottom"/>
  </Style>
  <Style ss:ID="s25">
   <NumberFormat ss:Format="@"/>
  </Style>
 </Styles>
 <Worksheet ss:Name="Timesheet">
  <Table ss:ExpandedColumnCount="3" ss:ExpandedRowCount="15"
➥x:FullColumns="1"
➥  x:FullRows="1">
   <Column ss:AutoFitWidth="0" ss:Width="70.5"/>
   <Column ss:AutoFitWidth="0" ss:Width="61.5"/>
   <Row>
    <Cell ss:StyleID="s22"><Data ss:Type="String">Employee
➥No.</Data></Cell>
    <Cell><Data ss:Type="Number">1234</Data></Cell>
   </Row>
   <Row>
    <Cell ss:StyleID="s22"><Data ss:Type="String">Surname:</Data></Cell>
    <Cell><Data ss:Type="String">Patterson</Data></Cell>
   </Row>
   <Row>
    <Cell ss:StyleID="s22"><Data ss:Type="String">First name:</Data></Cell>
    <Cell><Data ss:Type="String">Brian</Data></Cell>
   </Row>
   <Row ss:Index="5">
    <Cell ss:StyleID="s22"><Data ss:Type="String">Month</Data></Cell>
    <Cell ss:StyleID="s25"><Data ss:Type="String">January
```

```
↳2004</Data></Cell>
  </Row>
  <Row ss:Index="7">
   <Cell ss:StyleID="s23"><Data ss:Type="String">Day</Data></Cell>
   <Cell ss:StyleID="s22"><Data ss:Type="String">Std Hours</Data></Cell>
   <Cell ss:StyleID="s22"><Data ss:Type="String">Overtime</Data></Cell>
  </Row>
  <Row>
   <Cell ss:StyleID="s24"><Data ss:Type="Number">1</Data></Cell>
   <Cell><Data ss:Type="Number">7.5</Data></Cell>
   <Cell><Data ss:Type="Number">2</Data></Cell>
  </Row>
  <Row>
   <Cell ss:StyleID="s24"><Data ss:Type="Number">2</Data></Cell>
  </Row>
  <Row>
   <Cell ss:StyleID="s24"><Data ss:Type="Number">3</Data></Cell>
  </Row>
  <Row>
   <Cell ss:StyleID="s24"><Data ss:Type="Number">4</Data></Cell>
   <Cell><Data ss:Type="Number">7.5</Data></Cell>
   <Cell><Data ss:Type="Number">1</Data></Cell>
  </Row>
  <Row>
   <Cell ss:StyleID="s24"><Data ss:Type="Number">5</Data></Cell>
   <Cell><Data ss:Type="Number">6</Data></Cell>
  </Row>
  <Row>
   <Cell ss:StyleID="s24"><Data ss:Type="Number">6</Data></Cell>
   <Cell><Data ss:Type="Number">7.5</Data></Cell>
  </Row>
  <Row>
   <Cell ss:StyleID="s24"><Data ss:Type="Number">7</Data></Cell>
   <Cell><Data ss:Type="Number">7.5</Data></Cell>
  </Row>
  <Row>
   <Cell ss:StyleID="s24"><Data ss:Type="Number">8</Data></Cell>
   <Cell><Data ss:Type="Number">7.5</Data></Cell>
  </Row>
 </Table>
 <WorksheetOptions xmlns="urn:schemas-microsoft-com:office:excel">
  <Print>
   <ValidPrinterInfo/>
   <HorizontalResolution>600</HorizontalResolution>
   <VerticalResolution>0</VerticalResolution>
  </Print>
  <Selected/>
  <ProtectObjects>False</ProtectObjects>
  <ProtectScenarios>False</ProtectScenarios>
 </WorksheetOptions>
</Worksheet>
<Worksheet ss:Name="Sheet2">
 <WorksheetOptions xmlns="urn:schemas-microsoft-com:office:excel">
  <ProtectObjects>False</ProtectObjects>
  <ProtectScenarios>False</ProtectScenarios>
 </WorksheetOptions>
</Worksheet>
<Worksheet ss:Name="Sheet3">
 <WorksheetOptions xmlns="urn:schemas-microsoft-com:office:excel">
  <ProtectObjects>False</ProtectObjects>
  <ProtectScenarios>False</ProtectScenarios>
```

```
 </WorksheetOptions>
</Worksheet>
</Workbook>
```

Код XML для книги содержит данные, формулы и форматирование, но не содержит встроенных объектов, диаграмм и кода VBA.

> Дополнительная информация о поддержке XML в Excel 2003 доступна на сайте компании Microsoft по адресу `http://msdn.microsoft.com/Office/`.

Использование XSLT для преобразования XML

Рассматривая данные, соответствующие схеме XML-SS, можно заметить, что они выражаются в терминах книг, листов, таблиц, строк и ячеек, хотя в показанной ранее карточке учета рабочего времени данные представлены с точки зрения объектов `Timesheets`, `Employees`, `ReportingMonth` и `HoursWorked`. Перед отправкой данных карточки учета рабочего времени на Web-сервер необходимо преобразовать схему XML-SS в схему `Timesheet`. Эта задача выполняется с помощью XSLT или eXtensible Stylesheet Language: Transformations.

XSLT является декларативным языком, позволяющим применять таблицы стилей к исходному документу XML. Таблицы стилей выбирают из исходного кода XML определенные элементы данных в определенном порядке. В результате получается новый документ XML, соответствующий целевой схеме. В данном случае преобразование будет выглядеть следующим образом:

```
XML-SS + Собственный лист стилей = Timesheet XML
```

Ниже показан код XSLT для такого преобразования:

```
<?xml version="1.0"?>
<xsl:stylesheet version="1.0"
     xmlns:xsl="http://www.w3.org/1999/XSL/Transform"
     xmlns:xl="urn:schemas-microsoft-com:office:spreadsheet"
     xmlns:ss="urn:schemas-microsoft-com:office:spreadsheet"
    exclude-result-prefixes="xl ss">
<xsl:template match="/xl:Workbook/xl:Worksheet/xl:Table">
<TS:Timesheet xmlns:TS="urn:schemas_wrox_com:ExcelProgRef:Timesheet">
  <TS:Employee>
    <TS:FirstName>
      <xsl:value-of
↳select="xl:Row/xl:Cell[xl:NamedCell[@ss:Name='ForeName']]/xl:Data"/>
    </TS:FirstName>
    <TS:LastName>
      <xsl:value-of
↳select="xl:Row/xl:Cell[xl:NamedCell[@ss:Name='Surname']]/xl:Data"/>
    </TS:LastName>
    <TS:EmployeeNumber>
      <xsl:value-of
↳select="xl:Row/xl:Cell[xl:NamedCell[@ss:Name='EmpNo']]/xl:Data"/>
    </TS:EmployeeNumber>
  </TS:Employee>
  <TS:ReportingMonth>
    <TS:Year>
      <xsl:value-of
↳select="substring(xl:Row/xl:Cell[xl:NamedCell[@ss:Name='Month']]/xl
↳:Data,1,4)"/>
    </TS:Year>
```

```
    <TS:Month>
      <xsl:value-of
↳select="substring(xl:Row/xl:Cell[xl:NamedCell[@ss:Name='Month']]/xl
↳:Data,6,2)"/>
     </TS:Month>
    <TS:TotalStdHours>
      <xsl:value-of
↳select="xl:Row/xl:Cell[xl:NamedCell[@ss:Name='StdHours']]/xl:Data"/>
     </TS:TotalStdHours>
   </TS:ReportingMonth>
   <xsl:for-each select="xl:Row/xl:Cell[xl:NamedCell[@ss:Name='Days']]">
     <xsl:if
↳test="string(number(../xl:Cell[xl:NamedCell[@ss:Name='Hours']]/xl
↳:Data))!= 'NaN'">
      <TS:HoursWorked>
      <TS:Day>
        <xsl:value-of
↳select="../xl:Cell[xl:NamedCell[@ss:Name='Days']]/xl:Data"/>
        </TS:Day>
        <TS:StdHours>
          <xsl:value-of
↳select="../xl:Cell[xl:NamedCell[@ss:Name='Hours']]/xl:Data"/>
        </TS:StdHours>
        <xsl:choose>
            <xsl:when test="string(number
↳(../xl:Cell[xl:NamedCell[@ss:Name='Overtime']]/xl :Data))!='NaN'">
            <TS:Overtime>
              <xsl:value-of select="../xl:Cell[xl:NamedCell
↳[@ss:Name='Overtime']]/xl:Data"/>
            </TS:Overtime>
          </xsl:when>
          <xsl:otherwise><TS:Overtime/></xsl:otherwise>
        </xsl:choose>
      </TS:HoursWorked>
    </xsl:if>
  </xsl:for-each>
</TS:Timesheet>
</xsl:template>
</xsl:stylesheet>
```

Код XSLT может читать и человек. Главное, понять следующие правила.

- Только теги, которые начинаются с <xsl> и </xsl> являются элементами обработки. В них программе разбора XML сообщается, как выполнять преобразование данных.
- Все остальные теги копируются в исходящий поток.
- Следующее выражение находит ячейку с именем диапазона ForeName и возвращает элемент Data (то есть, его значение). В языке VBA эквивалентом такого преобразования является инструкция Range("ForeName").Value.

```
select="xl:Row/xl:Cell[xl:NamedCell[@ss:Name='ForeName']]/xl:Data"
```

- Нижеприведенное выражение запускает цикл For... Each для всех ячеек именованного диапазона Days. Эквивалентом этого выражения в языке VBA является оператор For Each oCell In Range("Days").Cells.

```
<xsl:for-each select="xl:Row/xl:Cell[xl:NamedCell[@ss:Name='Days']]">
```

- В схеме XML-SS существует древовидная структура Лист⇨Строки⇨Ячейки, поэтому родительским объектом для ячейки (../ в XSLT) является строка, а значит, эквивалентом выражения

```
select="../xl:Cell[xl:NamedCell[@ss:Name='Hours']]/xl:Data"/>
```

в языке VBA является

```
Application.Intersect(oCell.EntireRow,Range("Hours")).Value
```

- При трансформации удаляются дни, информация для которых не вводилась. Следующее выражение XSLT

```
test="string(number(...))!='NaN'"
```

эквивалентно функции VBA `IsNumeric()`.

Ссылка на этот код размещается в ячейке "XSLT" на листе "XSLT" (рис. 28.9).

Код VBA для преобразования и отправки кода Timesheet XML приложению на Web-сервере показан ниже. Для преобразования в данном случае используется программа разбора MSXML (эта программа разбора доступна в объектной библиотеке "Microsoft XML, v3.0").

```
' Преобразовать формат XML-SS в формат TimeSheet  XML
Sub TransformAndPostXML()

    ' Определить три экземпляра программы разбора MSXML


    ' Один экземпляр будет использоваться для исходного
    ' кода XML — в формате XML-SS
    Dim xmlSource As MSXML2.DOMDocument30

    ' Один экземпляр будет использоваться для преобразования XSLT
    Dim xmlXSLT As MSXML2.DOMDocument30

    ' Один экзмепляр будет использоваться для преобразования
    ' полученного кода XML в формат TimeSheet
    Dim xmlTimesheet As MSXML2.DOMDocument30

    ' Определить экземпляр объекта Internet Transfer Control
    ' для отправки кода XML
    Dim oInet As Inet
    Dim sTimesheetXML As String

    ' Создать новые экземпляры программы разбора MSXML
    Set xmlSource = New MSXML2.DOMDocument30
    Set xmlXSLT = New MSXML2.DOMDocument30
    Set xmlTimesheet = New MSXML2.DOMDocument30

    ' Загрузить исходный код XML с листа Timesheet
    xmlSource.loadXML Worksheets("TimeSheet").Range("A1:C40") _
                      .Value(xlRangeValueXMLSpreadsheet)

    ' Загрузить код XSLT с листа XSLT
    xmlXSLT.loadXML Worksheets("XSLT").Range("XSLT").Value

    ' Преобразовать схему XML-SS с помощью кода XSLT для получения
    ' кода TimeSheet XML
    xmlSource.transformNodeToObject xmlXSLT, xmlTimesheet
```

```
    ' Считать полученный код TimeSheet XML в переменную
    sTimesheetXML = xmlTimesheet.XML

    ' Создать новый экземпляр объекта Internet Transfer Control
    Set oInet = New Inet

    ' Отправить код XML на сервер
    oInet.Execute "http://localhost/MyPage.asp", "POST", sTimesheetXML

End Sub
```

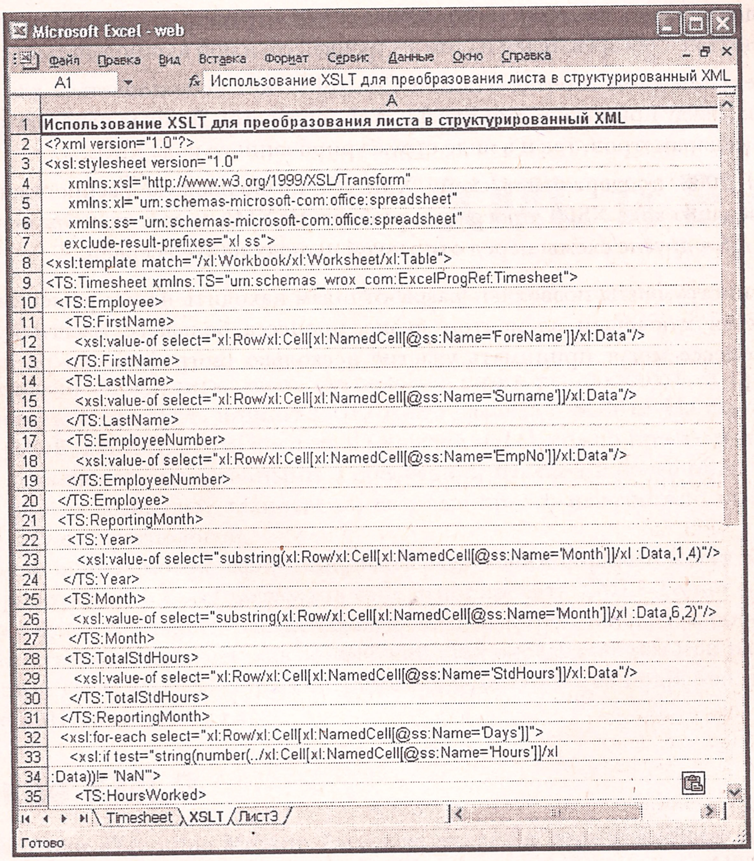


Рис. 28.9. Код XSLT для преобразования схемы документа

> Полное описание языка XSLT выходит за пределы тематики этой книги. Более подробная информация доступна в книге *XSLT Programmer's Reference*, Wrox Press, ISBN 1-861005-06-7.

Резюме

В Excel 2003 компания Microsoft предоставила разработчикам приложений Excel возможность использования сети Internet в качестве неотъемлемой части приложения.

- Книги могут открываться и сохранятся на Web-сервере, который поддерживает серверные расширения Front Page.
- Excel может открывать HTML-страницы в виде книг.
- Web-запросы могут использоваться для извлечения таблиц с данными с Web-страниц.
- Объектная библиотека Internet Explorer обеспечивает автоматизацию извлечения данных из Web-страницы без накладных расходов работы с книгой.
- Книги Excel могут сохраняться в виде Web-страниц с сохранением содержимого.
- Поддерживается создание интерактивных Web-страниц, которые обеспечивают взаимодействие с обозревателем на том же уровне, что и при взаимодействии с Excel.
- Объект Microsoft Internet Transfer Control может использоваться для обмена данными между приложениями Excel и Web-серверами. Данные передаются как в виде списка параметров, так и в виде структурированного кода XML.
- Excel 2003 поддерживает схему XML-SS, позволяющую распространять листы. С помощью кода XSLT программа разбора MSXML может преобразовывать схему XML-SS в другие схемы перед отправкой данных на Web-сервер.

Все эти инструменты позволяют разработчикам находить новые решения, в которых Excel является ключевым элементом бизнес-процесса большего масштаба. При этом в бизнес-процессе могут принимать участие несколько географически распределенных организаций.

29

XML и Excel

Иногда сложно определить, какие технологии стоит использовать, а какие можно проигнорировать. Кто-нибудь помнит DDE? А кто слышал о SGML? На самом деле эти технологии (DDE, к SGML это относится в меньшей мере) можно просто игнорировать, так как они являлись промежуточными и обеспечили переход к более подходящему решению. Но иногда возникают технологии, которые просто невозможно игнорировать. XML — одна из таких технологий.

Более точно вопрос можно сформулировать следующим образом: могут ли разработчики VBA-приложений в Excel позволить себе игнорировать XML? Точно сформулированный ответ будет выглядеть так: ненадолго. XML быстро становится определяющим языком в развитии сети Internet, как английский стал определяющим при развитии технологий. В этой главе язык XML рассматривается немного более подробно, включая использование XML совместно с Excel.

Что такое XML?

Язык XML или eXtensible Markup Language является подмножеством языка SGML (Standard Generalized Markup Language). XML — это язык разметки (как HTML), предоставляющий возможность хранения и описания данных. Самой важной особенностью языка XML является его самоописательная природа и текстовое представление. Так как XML описывает сам себя, он легко может переноситься в сетях, включая самую большую сеть — Internet. В сети Internet можно передавать не только текст, но все же в сетях TCP/IP текстовому формату отдается предпочтение.

Так как XML имеет текстовое представление, его очень просто передавать и обрабатывать. Так как в коде XML хранятся данные и элементы описания данных, это очень удобный формат для передачи информации. Поймите правильно, двоичные файлы (графика, видео, музыка) хорошо выполняют свою задачу, но XML станет универсальным языком в сети Internet в ближайшем будущем.

При первом знакомстве формат XML может напугать. Но минимальное знакомство с HTML позволит быстро привыкнуть к формату XML. В следующем листинге показан целый файл XML:

```
<?xml version="1.0" standalone="yes"?> <snapshot>
  <Recording>
    <Id>0</Id>
    <Count>0</Count>
    <When>2004-02-01T09:27:42.7096378-05:00</When>
    <InstanceId>0</InstanceId>
    <DealerHand>QD,JS</DealerHand>
    <PlayerHand>QC,10D</PlayerHand>
    <HandNumber>0</HandNumber>
    <CurrentBet>25</CurrentBet>
    <Money>500</Money>
    <PlayState>Push</PlayState>
    <Hints>Stand</Hints>
  </Recording>
</snapshot>
```

К сожалению, возможности привести здесь полное руководство по использованию языка XML нет, но издательство Wrox выпустило несколько хороших книг по XML, позволяющих стать профессионалом в работе с этим языком. При этом читать и писать код XML вручную не обязательно, так как большинство инструментов могут делать это автоматически. Текст XML имеет простой формат, используемый для передачи данных.

В этом листинге первый тег указывает на то, что текст является документом XML. Первая строка может иметь другую форму, но обычно первая строка документов XML очень похожа на приведенную здесь. Слова в <>, например, `<snapshot>`, являются тегами. Такие теги выступают в роли дескрипторов для описания типа данных. В определенной степени эти дескрипторы можно воспринимать как описания типов в объектно-ориентированном программировании. Внешние дескрипторы, например `<snapshot>`, определяют пространство имен (область действия имени). Вложенные дескрипторы, например `<Recording>`, могут рассматриваться как аналог определенного пользователем типа данных или класса. Таким образом, в пространстве имен или области видимости имен `<snapshot>` определен тип, который называется `<Recording>`. Наконец, самые внутренние элементы являются членами типов. Другими словами, объект `snapshot.Recording` содержит члены `Id`, `Count`, `When`, `InstanceId`, `DealerHand`, `PlayerHand`, `HandNumber`, `CurrentBet`, `Money`, `PlayState` и `Hints`.

Самым внешним элементом может быть `<BlackJackhand>`, но даже в приведенном виде достаточно простой файл XML содержит значительный объем информации о передаваемых данных. Имена элементов указываются явно вместе со значениями. Типы данных могут быть получены так же, как и значения. (XML поддерживает явное указание типа данных и не только.)

Что такое XSD?

XSD, или XML Schema Definition (определение схемы XML), является технологией описания документа XML без данных. Как и в случае технологий баз данных (а XML в некотором смысле является универсальным механизмом хранения данных), в технологии XSD существуют таблицы и схемы. В таблицах содержатся данные, а в схемах — описание данных, хранящихся в таблицах. При этом в файле XSD содержится схема, описывающая варианты, типы и структуру данных в соответствующем файле XML.

Если записать схему для XML-файлов, содержащих структуру сданных наборов карт в BlackJack, она будет выглядеть следующим образом.

```
<?xml version="1.0" ?>
<xs:schema id="snapshot" targetNamespace="http://tempuri.org/game.xsd"

xmlns:mstns="http://tempuri.org/game.xsd"
  xmlns="http://tempuri.org/game.xsd"
xmlns:xs="http://www.w3.org/2001/XMLSchema" xmlns:msdata="urn:schemas-
↳ microsoft-com:xml-msdata"
  attributeFormDefault="qualified" elementFormDefault="qualified">
  <xs:element name="snapshot" msdata:IsDataSet="true"
↳ msdata:EnforceConstraints="False">
    <xs:complexType>
      <xs:choice maxOccurs="unbounded">
          <xs:element name="Recording">
            <xs:complexType>
              <xs:sequence>
                <xs:element name="Id" type="xs:integer"
↳ minOccurs="0" />
                <xs:element name="Count" type="xs:integer"
↳ minOccurs="0" />
                <xs:element name="When" type="xs:dateTime"
↳ minOccurs="0" />
                <xs:element name="InstanceId" type="xs:integer"
↳ minOccurs="0" />
                <xs:element name="DealerHand" type="xs:string"
↳ minOccurs="0" />
                <xs:element name="PlayerHand" type="xs:string"
↳ minOccurs="0" />
                <xs:element name="HandNumber" type="xs:string"
↳ minOccurs="0" />
                <xs:element name="CurrentBet" type="xs:double"
↳ minOccurs="0" />
                <xs:element name="Money" type="xs:double"
↳ minOccurs="0" />
                <xs:element name="PlayState" type="xs:string"
↳ minOccurs="0" />
                <xs:element name="Hints" type="xs:string"
↳ minOccurs="0" />
              </xs:sequence>
            </xs:complexType>
          </xs:element>
      </xs:choice>
    </xs:complexType>
  </xs:element>
</xs:schema>
```

При этом схема выглядит сложнее, чем сами данные, но помните, что преимуществом такого формата является полное описание связанных со схемой данных, а также использование текстового представления. Кроме этого, применение текстового формата позволяет легко передавать файлы XSD по сети, что делает их неплохим общим знаменателем.

Что такое XMLSS?

В Excel схемы XSD непосредственно не применяются. Вместо этого Excel выполняет преобразование схем в формат XMLSS (XML Spreadsheet Schemas). В частности, Excel преобразовывает типы данных из схем в типы данных Excel и отображает XMLSS в панели задач XML Source. В результате в Excel схема представлена в виде Spreadsheet Schema в панели задач Источник XML (XML Source) (рис. 29.1).

Обратите внимание на панель задач Источник XML (XML Source) (для доступа к этой панели выберите пункт Данные⇨XML⇨Источник XML (Data⇨XML⇨XML Source)). На ней будет показана XML Spreadsheet Schema с дескрипторами, но без данных. Разница заключается в том, что файл XML может содержать 0 или больше элементов, определенных в схеме. Например, может быть только одна пара тегов `<snapshot></snapshot>`, но пара `<Recording></Recording>` может повторяться любое количество раз. Это неплохая метафора для представления файла схемы в виде определения класса, а файла XML — в виде экземпляра класса.

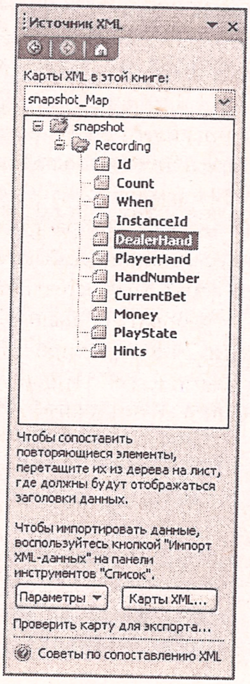


Рис. 29.1. Панель задач Источник XML

Импорт данных в формате XML

Реальным преимуществом XML и XSD является их использование в качестве общего знаменателя (в частности, они являются открытым стандартом, который никому не принадлежит). При этом все соглашаются, что такой формат является неплохим кандидатом на общий метод обмена данными. В результате все приложения, способные читать и записывать данные

в формате XML, могут общаться с другими приложениями, которые в состоянии работать с форматом XML. Возможность такого взаимодействия не зависит от языка, платформы, операционной системы, средств разработки и методов передачи данных.

Общий язык обмена данными для всех инструментов — это шаг в правильном направлении. Достаточно вспомнить, что несколько лет назад было практически невозможно обмениваться данными между Word, Excel и другими приложениями, особенно если эти приложения написаны независимыми разработчиками. Конечно, можно было использовать файлы с разделением полей запятыми, а механизм Automation поддерживал связывание кода непосредственно с Excel, но файлы с разделением запятыми не содержали информации о природе данных, то есть отсутствовала стандартизированная схема, а для применения механизма Automation требовалось обладать особой информацией об Excel. Стандартизация данных и схемы позволила каждому приложению экспортировать и импортировать XML. При этом для совместного использования данных не требуется информация о потребителях данных. То есть, для совместного применения данных с Excel приложение не должно использовать Excel VBA через механизм Automation. Для демонстрации гибкости этого подхода здесь используется недавно созданная простая игра.

Black Jack: гибкость данных

Игра BlackJack имеет несложные правила, но предоставляет широкий выбор стратегий игры. Хотя у нас и так получается неплохо играть, мы решили немного улучшить нашу игру. В результате была написана игра BlackJack (рис. 29.2). Она создана на базе инфраструктуры .NET. При создании игры были использованы разработки из теории вероятности, а также создан встроенный механизм записи, записывающий комбинации карт на руках ведущего и игрока, ставку и результат сдачи в файл XML. Создание механизма анализа игры и построение графа результатов может потребовать большего объема трудозатрат, чем хотелось бы, но при этом как минимум необходим механизм просмотра данных.

Именно для этого и нужна программа Excel. При создании игры, или при анализе хода игры для выигрыша дополнительной сотни в казино Вегаса, или при создании критического приложения для анализа пыли на Марсе, Excel можно использовать в качестве мощного аналитического инструмента. Если в качестве формата хранения данных использовать XML, эти данные можно будет легко импортировать в Excel.

Хотя игру BlackJack можно реализовать в Excel с помощью языка VBA и библиотеки `cards.dll` (которая предоставляется в составе Windows), эта глава посвящена другой теме. (Исходный код игры BlackJack доступен по адресу `http://www.softcocnepts.com/BlackJack/`.)

На рис. 29.3 показаны импортированные данные игры BlackJack, в которой начальная сумма увеличена с $500 и $2520. Это достаточно успешная игра, поэтому хотелось бы рассмотреть данные для получения представления о правильной стратегии. Можно заметить, что схема загружена из файла XML (показана в панели Источник XML), строки и столбцы преобразованы в списки, включая заголовки столбцов автофильтра и строки вставки (не показаны).

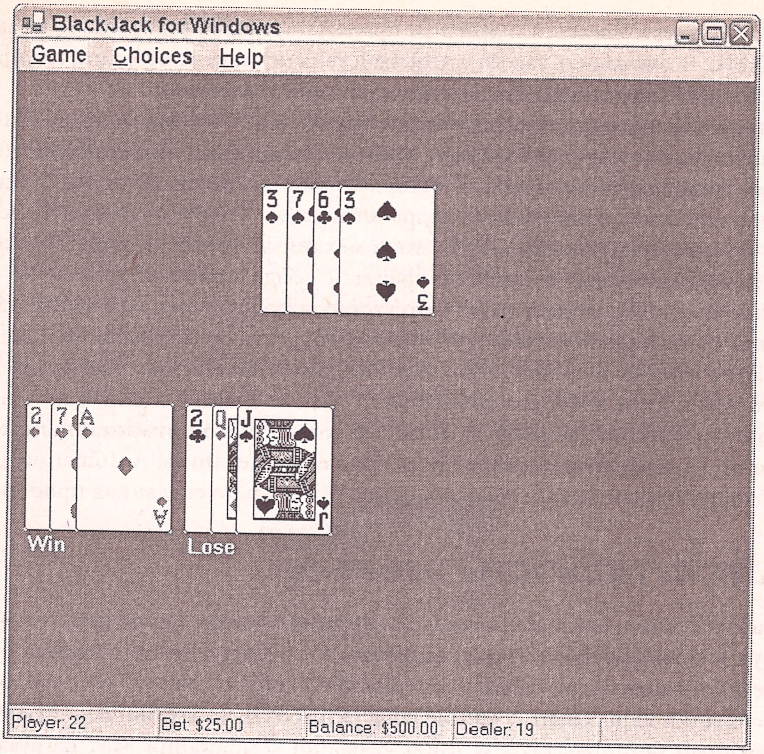


Рис. 29.2. Игра BlackJack

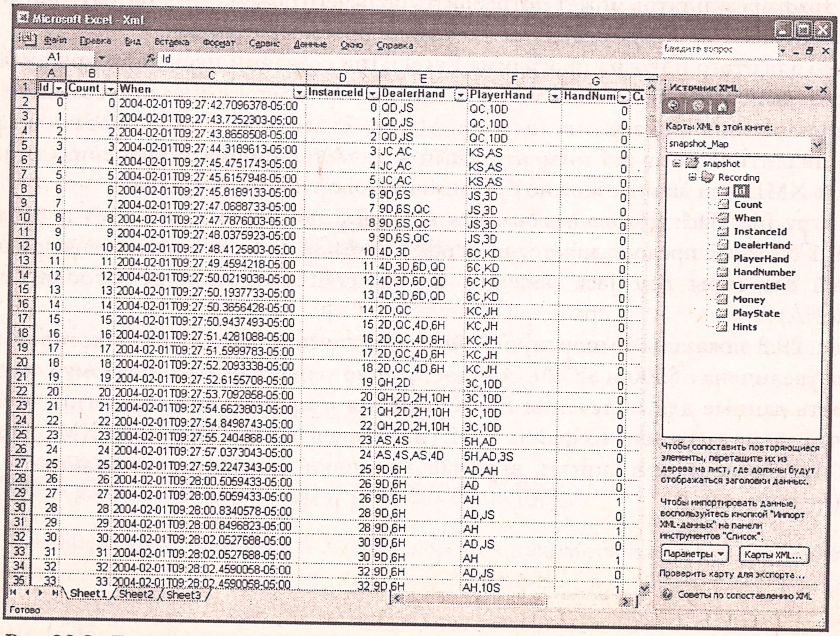


Рис. 29.3. Данные с результатами игры в BlackJack

Импорт файла XML

Реализация OpenXML вызывает метод OpenXML коллекции Application.Workbooks. Этот метод принимает три аргумента: исходный файл XML, файл XSLT (XML Style Sheet Transformation) и LoadOption. Файл XML содержит данные. Файл XSLT содержит документ XSL Transform (немного ниже этот документ будет рассматриваться более подробно), а параметр XlXmlLoadOption описывает способ обработки данных XML. Код из листинга эмулирует поведение пункта меню Данные⇨XML⇨Импорт (Data⇨XML⇨Import).

```
Public Sub OpenXML()   Dim aWorkbook As Workbook
    Set aWorkbook = Application.Workbooks.OpenXML( _
        Filename:="game.xml", LoadOption:=
↳XlXmlLoadOption.XlXmlLoadImportToList)
End Sub
```

В данном примере объявляется объект Workbook и вызывается метод Application.Workbooks.OpenXML. Этот метод возвращает книгу с импортированным кодом XML и извлеченной схемой Excel XML Spreadsheet Schema.

Что такое XSL и XSLT?

XSL — это аббревиатура, которая ссылается на язык разметки XML. В XSL разделяется представление данных (лист стилей) от фактических данных (файл XML). Например, передача параметра XlXMLLoadOption.XlXMLLoadImportToList приведет к форматированию данных XML в виде коллекции списков.

Теперь, когда есть представление о файлах XSL, возникает вопрос — что такое файл XSLT? Как было сказано ранее, XXSL содержит листы стилей для XML. В листах стилей описывается представление данных XML. Файл XSLT (XSL Transform) описывает форматирование и отображение данных XML. Для изучения XMLS и XSL можно порекомендовать второе издание книги издательства Wrox *XSLT: Programmer's Reference* Майкла Кея (Michael Kay).

В данном случае нужно знать две вещи: во-первых, для использования XML в Excel не обязательно изучать XML, XSL и XSLT, но это очень мощные технологии; во-вторых, рекомендуемая книга в 900 страниц, что, к сожалению, говорит о невозможности описания такой темы в данной главе. Путем пересмотра метода OpenXML и включения в него возможности использования файла XSL Transform, была показана разница между простым импортом данных XML и преобразованием и форматированием данных с помощью XSLT. В первом листинге показаны немного модифицированные первые несколько строк файла game.xml. В этих строках представлен файл XSLT, который используется для форматирования. Во втором листинге приведен новый файл преобразования game.xsl. В третьем листинге показана новая процедура OpenXML, используемая вместе с файлом XSLT.

```
<?xml version='1.0' ?>
<?xml-stylesheet type='text/xsl' href='game.xsl' ?>
<snapshot>
  <Recording>
    <Id>0</Id>
    <Count>0</Count>
    <When>2004-02-01T09:27:42.7096378-05:00</When>
    <InstanceId>0</InstanceId>
    <DealerHand>QD,JS</DealerHand>
    <PlayerHand>QC,10D</PlayerHand>
' ...  остаток файла намеренно не показан
```

Во второй строке полужирным шрифтом выделена новая информация. В результате для форматирования файла game.xml используется файл game.xsl. Помните, что форматирование отделяется от данных. При этом можно быстро поставить другой файл XSL и получить другое подмножество данных и полностью другое внешнее представление:

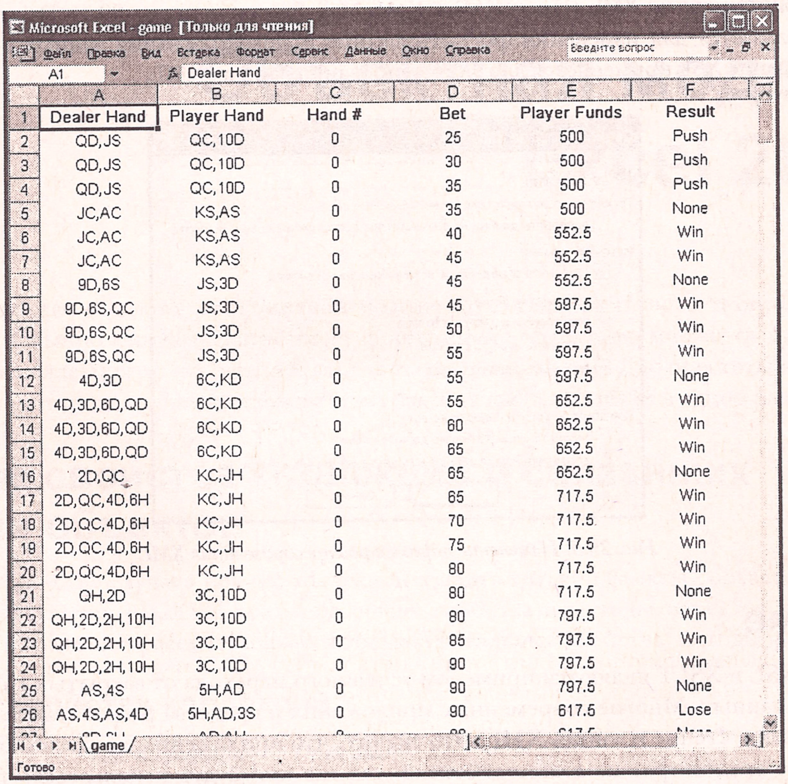
```
<xsl:stylesheet xmlns:xsl="http://www.w3.org/1999/XSL/Transform" _
version="1.0">
  <xsl:template match="/">
    <HTML>
     <body>
       <table>
        <COLGROUP WIDTH="100" ALIGN="CENTER"></COLGROUP>
           <COLGROUP WIDTH="100" ALIGN="CENTER"></COLGROUP>
        <COLGROUP WIDTH="100" ALIGN="CENTER"></COLGROUP>
           <COLGROUP WIDTH="100" ALIGN="CENTER"></COLGROUP>
        <COLGROUP WIDTH="100" ALIGN="CENTER"></COLGROUP>
           <COLGROUP WIDTH="100" ALIGN="CENTER"></COLGROUP>
        <COLGROUP WIDTH="100" ALIGN="CENTER"></COLGROUP>
           <TD><B>Dealer Hand</B></TD>
           <TD><B>Player Hand</B></TD>
           <TD><B>Hand #</B></TD>
           <TD><B>Bet</B></TD>
           <TD><B>Player Funds</B></TD>
           <TD><B>Result</B></TD>
           <xsl:for-each select="snapshot/Recording">
       <tr>
             <td><xsl:value-of select="DealerHand"/></td>
             <td><xsl:value-of select="PlayerHand"/></td>
         <td><xsl:value-of select="HandNumber"/></td>
             <td><xsl:value-of select="CurrentBet"/></td>
             <td><xsl:value-of select="Money"/></td>
             <td><xsl:value-of select="PlayState"/></td>
           </tr>
           </xsl:for-each>
      </table>
    </body>
  </HTML>
  </xsl:template>
</xsl:stylesheet>
```

В файле game.xsl содержится информация о форматировании. Для описания внешнего вида информации принято применять код HTML. Как показано в предыдущем листинге, для форматирования данных используется файл HTML с выделением заголовков столбцов полужирным шрифтом и размещением данных в таблице HTML. В результате данные будут выглядеть, как Web-страница. Полужирный шрифт для каждого оператора XSLT определяется только для выделения конкретных столбцов в файле game.xml. Кроме этого, можно создать более сложный файл XSLT, в котором над данными будут выполняться арифметические операции, добавляющие новые данные к статическим данным из файла game.xml.

Наконец, ниже приводится пересмотренная функция OpenXML. В следующем листинге показано, что параметр XlXmlLoadOption не используется. Вместо него применяется динамический массив Array(1). С помощью этого аргумента определяется используемый лист стилей для файла .XML. Например, если бы вторая строка выглядела как <?xml-stylesheet type='text/xsl' href='game2.xsl'?> и в качестве параметра передавался бы Array(2), то для форматирования данных использовался бы файл game2.xsl. Вот новый вариант подпрограммы OpenXML:

```
Public Sub OpenXML()
    Dim path As String
    path = ThisWorkbook.path + "\game.xml"
    Dim aWorkbook As Workbook
    Set aWorkbook = Application.Workbooks.OpenXML(path, Array(1))
End Sub
```

Хотя в эту главу нельзя втиснуть 900 страниц по XML, XSL и XSLT, из текста данной главы можно понять, что XML, кроме простого метода передачи данных по сети и между приложениями, поддерживает использование отдельного файла для форматирования и преобразования, что позволяет хранить одну копию данных и использовать несколько вариантов представления. На рис. 29.4 показан внешний вид динамической страницы в Excel.



	A	B	C	D	E	F
1	Dealer Hand	Player Hand	Hand #	Bet	Player Funds	Result
2	QD,JS	QC,10D	0	25	500	Push
3	QD,JS	QC,10D	0	30	500	Push
4	QD,JS	QC,10D	0	35	500	Push
5	JC,AC	KS,AS	0	35	500	None
6	JC,AC	KS,AS	0	40	552.5	Win
7	JC,AC	KS,AS	0	45	552.5	Win
8	9D,6S	JS,3D	0	45	552.5	None
9	9D,6S,QC	JS,3D	0	45	597.5	Win
10	9D,6S,QC	JS,3D	0	50	597.5	Win
11	9D,6S,QC	JS,3D	0	55	597.5	Win
12	4D,3D	6C,KD	0	55	597.5	None
13	4D,3D,6D,QD	6C,KD	0	55	652.5	Win
14	4D,3D,6D,QD	6C,KD	0	60	652.5	Win
15	4D,3D,6D,QD	6C,KD	0	65	652.5	Win
16	2D,QC	KC,JH	0	65	652.5	None
17	2D,QC,4D,6H	KC,JH	0	65	717.5	Win
18	2D,QC,4D,6H	KC,JH	0	70	717.5	Win
19	2D,QC,4D,6H	KC,JH	0	75	717.5	Win
20	2D,QC,4D,6H	KC,JH	0	80	717.5	Win
21	QH,2D	3C,10D	0	80	717.5	None
22	QH,2D,2H,10H	3C,10D	0	80	797.5	Win
23	QH,2D,2H,10H	3C,10D	0	85	797.5	Win
24	QH,2D,2H,10H	3C,10D	0	90	797.5	Win
25	AS,4S	5H,AD	0	90	797.5	None
26	AS,4S,AS,4D	5H,AD,3S	0	90	617.5	Lose

Рис. 29.4. Представление страницы в Excel

Экспорт листа в файл XML

Предположим, что в рассматриваемую версию данных внесены изменения, например, удалены конфиденциальные записи, а полученные данные необходимо экспортировать в новый файл XML для передачи клиентам. Для экспорта данных в новый файл можно воспользоваться методом `Workbook.SaveAsXMLData`.

В следующем листинге показан код для экспорта файла `game.xml` с помощью отображения XMLSS, созданного в Excel. Новый файл сохраняется под именем `newgame.xml`.

```
Public Sub ExportXML()
    Dim map As XmlMap
    Set map = ThisWorkbook.XmlMaps("snapshot_Map")
    Dim newPath As String
    newPath = ThisWorkbook.path + "\newgame.xml"
    If (map.IsExportable) Then
        Call ThisWorkbook.SaveAsXMLData(newPath, map)
    Else
        MsgBox " Невозможно использовать " & map.Name & _
               " для экспорта данных"
    End If
End Sub
```

Принятое по умолчанию имя отображения можно получить в диалоговом окне Данные⇨XML⇨Свойства карты XML (Data⇨XML⇨XML Map Properties) (рис. 29.5). Это диалоговое окно используется для изменения форматирования и преобразования данных в Excel, как преобразование XSL применяется для форматирования данных XML.

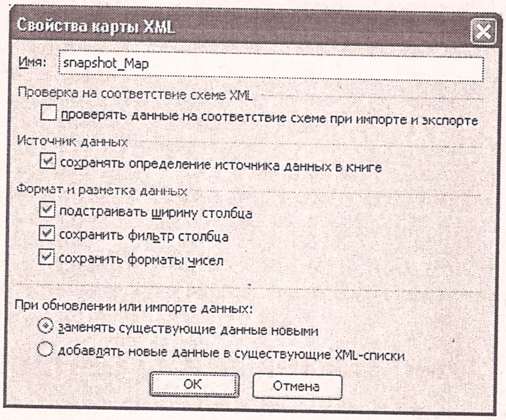


Рис. 29.5. Изменение параметров преобразования XML

Резюме

XML, XSL и XSLT являются примером успешного перехода от закрытых к открытым форматам данных. Многие современные приложения и средства разработки изначально поддерживают чтение и запись XML. Это значит, что приложения в будущем потребуют все меньше трудозатрат на взаимодействие, так как они будут использовать понятный всем общий стандарт.

Разработчики VBA-приложений в Excel могут читать данные XML от других приложений без создания специального кода. Одним из таких приложений является SharePoint (описывается в главе 21). Службы SharePoint используют XML для передачи данных, позволяя публиковать списки в сети Internet.

Технологии форматирования и преобразования данных (XSL и XSLT) позволяют отделять данные от форматирования и структуры. В результате один экземпляр данных может иметь несколько различных представлений.

A

Объектная модель Excel 2003

Большинство объектов из объектной модели Excel связаны с соответствующими коллекциями. Обычно имя коллекции имеет форму имени объекта во множественном числе. Например, коллекция `Worksheets` хранит объекты `Worksheet`. Для простоты изложения объекты и соответствующие коллекции будут рассматриваться вместе в одном разделе.

Общие свойства коллекций и связанных с ними объектов

В большинстве случаев объект коллекции хранит только объекты соответствующего типа. В следующих разделах показаны общие свойства и методы объектов коллекций. В каждом из разделов рассматриваются только уникальные свойства, методы и события.

Общие свойства коллекций

Имя	*Тип возвращаемого значения*	*Описание*
`Application`	`Application`	Только чтение. Возвращает ссылку на приложение, владеющее текущим объектом. В данном случае это Excel
`Count`	`Long`	Только чтение. Возвращает количество объектов в коллекции
`Creator`	`Long`	Только чтение. Возвращает число типа `Long`, описывающее место создания объекта (в Excel или за его пределами)
`Parent`	`Object`	Объект `Parent`, который является владельцем объекта коллекции. Например, свойство `Workbook.Parent` возвращает ссылку на объект `Application`

Общие свойства объектов

Объекты также обладают общими свойствами. Для избежания избыточности ниже перечислены общие свойства и методы всех объектов. Эти свойства будут упоминаться в описании объектов, но определение свойств приводится только здесь.

Имя	*Тип возвращаемого значения*	*Описание*
Application	Application	Только чтение. Возвращает ссылку на объект приложения, владеющий данным объектом (в данном случае это Excel)
Creator	Long	Только чтение. Возвращает число типа Long, которое описывает место создания объекта (в Excel или за его пределами)
Parent	Object	Только чтение. Объект-владелец текущего объекта. Например, свойство Characters.Parent может возвращать ссылку на объект Range, так как объект Range может быть одним из владельцев объекта Characters

Объекты Excel, их свойства, методы и события

Объекты перечислены в алфавитном порядке. Для каждого из них предоставляется общее описание и возможные родительские объекты. После этого в табличном формате приводится список всех свойств, методов и событий объекта. В последнем разделе описания каждого объекта представлены примеры кода с использованием объекта.

Объект Addin и коллекция Addins

В коллекции Addins содержатся все объекты Addin, доступные в Excel. Каждый объект Addin соответствует надстройке из списка в диалоговом окне Надстройки (Addin) (для открытия этого окна выберите пункт меню Сервис⇨Надстройки (Tools⇨Addin)). Перед использованием в текущем сеансе каждая надстройка должна быть установлена (Addin.Installed = True). В качестве примера объекта Addin для Excel можно привести Analysis Toolpak, MS Query Addin и Conditional Sum Wizard.

Метод Add коллекции Addins позволяет добавлять в коллекцию объекты Addin. Этот метод требует указания параметра FileName (обычно это имя файла с расширением XLL или XLA). Свойство Count коллекции Addins возвращает количество надстроек, которые доступны в текущем сеансе Excel.

Общие свойства объектов Addin

Определения свойств Application, Creator и Parent приводились в начале приложения.

Свойства объекта Addin

Имя	*Тип возвращаемого значения*	*Описание*
`CLSID`	`String`	Только чтение. Возвращает уникальный идентификатор объекта `Addin`
`FullName`	`String`	Только чтение. Возвращает полный путь и имя файла для соответствующего объекта `Addin`
`Installed`	`Boolean`	Устанавливает/возвращает возможность использования объекта `Addin` в текущем сеансе
`Name`	`String`	Только чтение. Возвращает имя файла для объекта `Addin`
`Path`	`String`	Только чтение. Возвращает полное имя файла для объекта `Addin`
`Title`	`String`	Только чтение. Это скрытое свойство возвращает строку, отображаемую в диалоговом окне Диспетчера надстроек (Addin Manager)

Пример: Объект Addin и коллекция Addins

В этом примере предполагается, что установлена надстройка Analysis Toolpak:

```
Sub UseAnalysisToolpack()
   Dim oAddin As AddIn
   ' Удостовериться, что надстройка Analysis Toolpak установлена
   For Each oAddin In AddIns
      If oAddin.Name = "ANALYS32.XLL" Then
         oAddin.Installed = True
      End If
   Next
End Sub
```

Обратите внимание, что вместо перебора коллекции `Addins` можно обратиться к интерактивному справочному руководству и воспользоваться следующим кодом:

```
AddIns("Analysis Toolpak").Installed = True
```

К сожалению, такой подход не работает для неанглийских языковых версий, если имя надстройки было локализовано.

Объект Adjustments

Объект `Adjustments` содержит коллекцию чисел, которые управляют "маркерами" изменения размера родительского объекта `Shape`. Каждый объект `Shape` имеет до 8 таких маркеров. Каждый маркер может быть связан с одним или двумя параметрами, в зависимости от возможности перемещения в горизонтальном или вертикальном направлении. Значения параметров находятся в пределах от 0 до 1 и измеряются в процентах. Абсолютное значение 100% изменения параметра определяется конкретным объектом `Shape`.

Общие свойства объекта Adjustments

Определения свойств `Application`, `Creator` и `Parent` приводились в начале приложения.

Свойства объекта Adjustments

Имя	Тип возвращаемого значения	Описание
Count	Long	Только чтение. Возвращает количество параметров, связанных с родительским объектом Shape
Item	Single	Параметры: Index As Long. Устанавливает или возвращает значение параметра или параметров, на которые указывает параметр Index

Пример: объект Adjustments

В этом примере на листе рисуется черная стрелка. После этого выполняется модификация формы наконечника стрелки.

```
Sub AddArrow()
    Dim oShp As Shape

    ' Добавить стрелку на лист
    Set oShp = ActiveSheet.Shapes.AddShape( _
                    msoShapeRightArrow, 10, 10, 100, 50)

    ' Разместить "наконечник" стрелки на расстоянии 30% от начала в ширину
    ' и разместить "древко" на расстоянии 40% от начала в высоту
    oShp.Adjustments(1) = 0.3      ' Слева направо
    oShp.Adjustments(2) = 0.4      ' Сверху вниз
End Sub
```

Объект AllowEditRange и коллекция AllowEditRanges

Объект AllowEditRange описывает диапазон ячеек на листе, которые можно редактировать при установленной защите. Каждый объект AllowEditRange может устанавливать права доступа для любого количества пользователей в сети и использовать отдельные пароли.

Применяя эту возможность, обратите внимание на свойство Locked объекта Range. При разблокировании ячеек и защите листа любой пользователь получает доступ к этим ячейкам вне зависимости от объектов AllowEditRange. При блокировании всех ячеек объекта AllowEditRange любой пользователь все еще может редактировать ячейки, пока не будет назначен пароль или не появятся новые пользователи и им не будет запрещен доступ без пароля.

Коллекция AllowEditRanges содержит все объекты AllowEditRange, которые можно редактировать на защищенном листе. Дополнительная информация приводится в разделе, посвященном объекту AllowEditRange.

Свойства коллекции AllowEditRanges

Имя	Тип возвращаемого значения	Описание
Count	Long	Только чтение. Возвращает количество объектов AllowEditRange, доступных в области
Item	AllowEditRange	Параметр: Index As Variant. Возвращает один объект AllowEditRange из коллекции AllowEditRanges

Методы коллекции AllowEditRanges

Имя	*Тип возвращаемого значения*	*Параметры*	*Описание*
Add	AllowEditRange	Title As String, Range As Range, [Password]	Добавляет объект AllowEditRange в коллекцию AllowEditRanges

Свойства объекта AllowEditRange

Имя	*Тип возвращаемого значения*	*Описание*
Range	Range	Возвращает подмножество диапазонов, которые можно редактировать на защищенном листе
Title	String	Возвращает или устанавливает заголовок Web-страницы при сохранении документа в виде Web-страницы
Users	UserAccessList	Возвращает список пользователей, которым разрешен доступ к защищенному диапазону листа

Методы объекта AllowEditRange

Имя	*Тип возвращаемого значения*	*Параметры*	*Описание*
ChangePassword		Password As String	Устанавливает пароль для диапазона, который можно редактировать на защищенном листе
Delete			Удаляет объект
Unprotect		[Password]	Снимает защиту с листа или книги

Пример: объект AllowEditRange

Следующая подпрограмма перебирает список имен диапазонов на листе и добавляет объект AllowEditRange для каждого диапазона, имя которого начинается с "pc". Кроме этого, подпрограмма запрещает доступ к диапазону pcNetSales всем, кроме одного пользователя, который может редактировать диапазон после ввода пароля.

```
Sub CreateAllowRanges()

    Dim lPos As Long
    Dim nm As Name
    Dim oAllowRange As AllowEditRange
    Dim sName As String

    With wksAllowEditRange
        ' Перебрать все имена диапазонов уровня листа
        For Each nm In .Names
            ' Сохранить имя
            sName = nm.Name

            ' Найти положение символа "!"
            lPos = InStr(1, sName, "!", vbTextCompare)

            ' Если символ "!" найден...
```

```
            If lPos > 0 Then
                ' Если сразу после восклицательного знака
                ' встречается последовательность
                ' "pc", создать объект AllowEditRange
                If Mid(sName, lPos + 1, 2) = "pc" Then
                    ' Проверить блокировку ячеек
                    ' Разблокирование ячеек предоставит доступ любому
                    ' пользователю
                    nm.RefersToRange.Locked = True

                    ' Извлечь ссылку на лист (включая "!")
                    ' из имени диапазона
                    sName = Right(sName, Len(sName) - lPos)

                    ' Создать объект AllowEditRange
                    ' Перед этим удалить уже существующий объект
                    On Error Resume Next
                        Set oAllowRange = Nothing
                        Set oAllowRange = _
.Protection.AllowEditRanges(sName)
                    On Error GoTo 0
                    If Not oAllowRange Is Nothing Then oAllowRange.Delete
                    Set oAllowRange = _
.Protection.AllowEditRanges.Add(sName, nm.RefersToRange)
                    ' Если это диапазон "Sales"
                    If sName = "pcNetSales" Then
                        ' Добавить пароль, после этого
                        ' добавить пользователя и запретить пользователю
                        ' редактировать диапазон без ввода пароля
                        oAllowRange.ChangePassword "pcnsw"
                        oAllowRange.Users.Add "RCR\AgamaOffice", False
                    End If
                End If
            End If
        Next nm
    End With

End Sub
```

Объект Application

Объект Application является корневым объектом объектной модели Excel. Доступ ко всем объектам из объектной модели возможен только через объект Application. При этом многие объекты доступны глобально, например свойство ActiveSheet объекта Application. Это значит, что доступ к активному листу можно получить двумя способами: через свойство Application.ActiveSheet и через ActiveSheet.

Объект Application содержит большинство атрибутов уровня приложения, которые можно устанавливать через диалоговое окно Сервис⇨Параметры (Tools⇨Options) в Excel. Например, свойство DefaultFilePath является эквивалентом полю Размещение файла по умолчанию (Default File Location) на вкладке Общие (General) диалогового окна Параметры (Options).

Большинство свойств и методов объекта Application имеют соответствующие эквиваленты в диалоговом окне Параметры (Options).

Кроме того, объект Application используется для автоматизации доступа к Excel из других приложений, например Word. Функции CreateObject и GetObject, а также

ключевое слово New используются для создания нового экземпляра объекта Excel Application из другого приложения. Дополнительная информация по автоматизации доступа из другого приложения приводится в главе 15.

Кроме этого, объект Application предоставляет ряд событий. Но события объекта Application не доступны автоматически. Для использования событий объекта Application необходимо выполнить следующие три операции.

Сначала следует создать новый модуль класса, например, cAppObject, и объявить в классе объектную переменную с квалификатором доступа Public, например, AppExcel. Эта объектная переменная будет обеспечивать обработку событий. Объявление переменной выглядит следующим образом:

```
Public WithEvents AppExcel As Excel.Application
```

После этого события объекта Application становятся доступны в классе объектной переменной AppExcel. При этом в классе реализуются соответствующие процедуры обработки событий. Например, если необходимо, чтобы при каждой активизации листа выводилось сообщение, напишите следующий код:

```
Private Sub AppExcel_SheetActivate(ByVal Sh As Object)
   ' Вывести имя листа
   MsgBox "Только что был активизирован лист " & Sh.Name & "."
End Sub
```

Наконец, с помощью текущего объекта Application создайте экземпляр класса в процедуре стандартного модуля.

```
Private App As New cAppObject ' класс с показанными выше фрагментами кода
Sub AttachEvents()
   Set App.AppExcel = Application
End Sub
```

Свойство EnableEvents объекта Application должно быть установлено в значение True. В противном случае события не будут генерироваться.

Общие свойства объекта Application

Определения свойств Application, Creator и Parent приводились в начале приложения.

Свойства объекта Application

Имя	*Тип возвращаемого значения*	*Описание*
ActiveCell	Range	Только чтение. Возвращает ячейку активного листа, в которой расположен курсор
ActiveChart	Chart	Только чтение. Возвращает выделенную диаграмму из активной книги. Если в данный момент не выделено ни одной диаграммы, свойство ничего не возвращает
ActivePrinter	String	Устанавливает/возвращает имя текущего принтера
ActiveSheet	Object	Только чтение. Возвращает активный лист активной книги

Имя	*Тип возвращаемого значения*	*Описание*
ActiveWindow	Window	Только чтение. Возвращает выделенное окно Excel
ActiveWorkbook	Workbook	Только чтение. Возвращает активную книгу
AddIns	AddIns	Только чтение. Возвращает коллекцию надстроек, доступных в Excel в данный момент
AlertBefore-Overwriting	Boolean	Устанавливает/возвращает необходимость выдачи сообщения при каждой попытке перезаписи непустой ячейки в результате операции перетаскивания и отпускания
AltStartupPath	String	Устанавливает/возвращает папку с альтернативными файлами запуска Excel
AnswerWizard	AnswerWizard	Только чтение. Возвращает объект, поддерживающий управление мастером ответов
ArbitraryXML-SupportAvailable	Boolean	Возвращает доступность функций работы с XML
AskToUpdate-Links	Boolean	Устанавливает/возвращает необходимость выдачи запроса на обновление связей при открытии содержащей записи книги
Assistant	Assistant	Только чтение. Возвращает объект, поддерживающий управление Помощником Office
AutoCorrect	AutoCorrect	Только чтение. Возвращает объект, поддерживающий управление возможностями автозамены в Excel
AutoFormatAs-YouTypeReplace-Hyperlinks	Boolean	Устанавливает/возвращает возможность автоматического форматирования/создания гиперссылок при вводе текста
Automation-Security	MsoAutomation-Security	Устанавливает/возвращает уровень безопасности макросов при программном открытии файла в Excel. Этот параметр не зависит от параметров безопасности макросов в диалоговом окне Безопасность (Security), доступном через меню Сервис⇨Макрос (Tools⇨Macros), хотя константа `msoAutomationSecurityByUI` заставляет свойство использовать значение параметра из диалогового окна
AutoPercent-Entry	Boolean	Устанавливает/возвращает необходимость добавления символа % при вводе числа в ячейку с выбранным Процентным форматом чисел
AutoRecover	AutoRecover	Устанавливает/возвращает параметры автовосстановления, например `Path` и `Time`
Build	Long	Только чтение. Возвращает точный номер версии Excel
CalculateBefore-Save	Boolean	Устанавливает/возвращает необходимость пересчета книг перед сохранением на диске. Предполагается, что автоматический пересчет формул отключен (свойство `Calculation`)

Имя	*Тип возвращаемого значения*	*Описание*
Calculation	XlCalculation	Устанавливает/возвращает режим пересчета формул: автоматический, ручной или полуавтоматический
Calculation-InterruptKey	XlCalculation-InterruptKey	Устанавливает/возвращает клавишу, которая может прервать операцию пересчета в Excel
Calculation-State	XlCalculation-State	Только чтение. Указывает текущее состояние расчетов в Excel: в процессе, в очереди или завершены
Calculation-Version	Long	Только чтение. Возвращает версию Excel и версию механизма расчетов, которые использовались при последнем сохранении файла
Caller	Variant	Только чтение. Параметры: [Index]. Возвращает информацию, описывающую источник вызова текущего кода Visual Basic (например, функцию ячейки или событие документа)
CanPlaySounds	Boolean	Только чтение. Возвращает возможность воспроизведения звуковых уведомлений в Excel. Это свойство не используется начиная с Excel 2000
CanRecordSounds	Boolean	Только чтение. Возвращает возможность записи звуковых уведомлений в Excel. Это свойство не используется начиная с Excel 2000
Caption	String	Устанавливает/возвращает заголовок основного окна Excel
CellDragAndDrop	Boolean	Устанавливает/возвращает возможность перетаскивания ячеек
Cells	Range	Только чтение. Возвращает все ячейки активного листа
Charts	Sheets	Только чтение. Возвращает все диаграммы активной книги
ClipboardFormats	Variant	Только чтение. Параметры: [Index]. Возвращает массив значений формата (XlClipboardFormat), находящегося в буфере обмена в данный момент
Columns	Range	Только чтение. Возвращает все столбцы активного листа
COMAddIns	COMAddIns	Только чтение. Возвращает коллекцию установленных надстроек COM
CommandBars	CommandBars	Только чтение. Возвращает коллекцию командных панелей, доступных в Excel
CommandUnder-lines	XlCommand-Underlines	Устанавливает/возвращает способ подчеркивания команд в Excel. Используется только на платформе Macintosh
ConstrainNumeric	Boolean	Устанавливает/возвращает необходимость распознавания ручного ввода только чисел и знаков пунктуации. Используется на платформе Windows, поддерживающей рукописный ввод

Имя	*Тип возвращаемого значения*	*Описание*
`ControlCharacters`	`Boolean`	Устанавливает/возвращает отображение управляющих символов в языках с записью справа налево. (Для этого должна быть установлена поддержка языка)
`CopyObjectsWithCells`	`Boolean`	Устанавливает/возвращает возможность вырезания, копирования и сортировки объектов (например, встроенных объектов) одновременно с данными в ячейках
`Cursor`	`XlMousePointer`	Устанавливает/возвращает указатель мыши, который отображается в Microsoft Excel
`CursorMovement`	`Long`	Устанавливает/возвращает тип используемого курсора: визуальный или логический
`CustomListCount`	`Long`	Только чтение. Возвращает количество созданных и встроенных списков, используемых в Excel (например, Понедельник, Вторник, Среда...)
`CutCopyMode`	`XlCutCopyMode`	Устанавливает/возвращает текущее состояние операции вырезания или копирования
`DataEntryMode`	`Long`	Устанавливает/возвращает возможность редактирования заблокированных ячеек (`xlOff` позволяет редактировать, `xlOn` позволяет редактировать только незаблокированные ячейки, `xlStrict` позволяет редактировать только те незаблокированные ячейки, которые нельзя отменить с помощью нажатия клавиши <Escape>)
`DDEAppReturnCode`	`Long`	Только чтение. Возвращает результат (подтверждение/ошибку) последнего сообщения DDE, отправленного Excel
`DefaultFilePath`	`String`	Устанавливает/возвращает принятую по умолчанию папку, которая используется при открытии файлов
`DefaultSaveFormat`	`XlFileFormat`	Устанавливает/возвращает принятый по умолчанию формат файла, который используется при сохранении
`DefaultSheetDirection`	`Long`	Устанавливает/возвращает направление нового листа в Excel
`DefaultWebOptions`	`DefaultWebOptions`	Только чтение. Возвращает объект, позволяющий управлять параметрами, доступными в диалоговом окне Параметры веб-документа (Web Options)
`DecimalSeparator`	`String`	Устанавливает/возвращает символ, который используется в виде десятичного разделителя. Это глобальный параметр, оказывающий влияние на все открытые книги. Для сброса собственных разделителей используется оператор `Application.UseSystemSeparators = True`
`Dialogs`	`Dialogs`	Только чтение. Возвращает коллекцию всех встроенных диалоговых окон

Имя	*Тип возвращаемого значения*	*Описание*
`DisplayAlerts`	`Boolean`	Устанавливает/возвращает необходимость вывода типичных сообщений Excel (например, “Сохранять изменения в книге?”). Сообщения можно отключать. В таком случае всегда будет выбираться ответ, принятый по умолчанию
`DisplayClipboardWindow`	`Boolean`	Устанавливает/возвращает необходимость вывода окна буфера обмена. Используется в Microsoft Office Macintosh Edition
`DisplayCommentIndicator`	`XlCommentDisplayMode`	Устанавливает/возвращает режим отображения комментариев и индикаторов в ячейках
`DisplayDocumentActionTaskPane`	`Boolean`	Устанавливается в значение `True` для отображения панели задач Действия с документами (Document Actions)
`DisplayExcel4Menus`	`Boolean`	Устанавливает/возвращает необходимость отображения меню Excel 4.0
`DisplayFormulaBar`	`Boolean`	Устанавливает/возвращает необходимость отображения панели Формула
`DisplayFullScreen`	`Boolean`	Устанавливает/возвращает необходимость работы в полноэкранном режиме
`DisplayFunctionToolTips`	`Boolean`	Устанавливает/возвращает режим отображения подсказок для аргументов функций при вводе функции в ячейку
`DisplayInsertOptions`	`Boolean`	Устанавливает/возвращает необходимость вывода раскрывающегося списка Параметры вставки (Insert Options) возле диапазона после вставки ячеек, строк или столбцов
`DisplayNoteIndicator`	`Boolean`	Устанавливает/возвращает необходимость отображения индикатора комментария в верхнем правом углу ячейки
`DisplayPasteOptions`	`Boolean`	Устанавливает/возвращает необходимость отображения кнопки Параметры вставки (Paste Options) возле диапазона после операции вставки. Этот параметр доступен в пакете Office XP, а значит, оказывает влияние на все остальные приложения Office, в которых используется эта возможность
`DisplayRecentFiles`	`Boolean`	Устанавливает/возвращает необходимость отображения списка последних открытых файлов в меню Файл (File)
`DisplayScrollBars`	`Boolean`	Устанавливает/возвращает необходимость отображения полос прокрутки для всех открытых книг в текущем сеансе
`DisplayStatusBar`	`Boolean`	Устанавливает/возвращает необходимость отображения строки состояния

Имя	*Тип возвращаемого значения*	*Описание*
EditDirectlyInCell	Boolean	Устанавливает/возвращает возможность модификации существующего текста непосредственно в ячейке. Обратите внимание, что такая перезапись текста в ячейке возможна в любом случае
EnableAnimations	Boolean	Устанавливает/возвращает необходимость анимации удаления и добавления ячеек, строк и столбцов
EnableAutoComplete	Boolean	Устанавливает/возвращает текущее состояние возможности автодополнения (AutoComplete)
EnableKeyCancelKey	XlEnableCancelKey	Устанавливает/возвращает реакцию макроса Excel на попытку пользователя прервать выполнение макроса (например, с помощью комбинации клавиш <Ctrl+Break>). Этот параметр можно использовать для отключения прерываний со стороны пользователя, перенаправления прерываний обработчику ошибок или для остановки выполнения кода (этот вариант принят по умолчанию). Использовать осторожно
EnableEvents	Boolean	Устанавливает/возвращает необходимость генерации событий для объектов из объектной модели Excel, поддерживающих события
EnableSound	Boolean	Устанавливает/возвращает возможность использования звуков в Excel
ErrorCheckingOptions	ErrorCheckingOptions	Устанавливает/возвращает свойства проверки ошибок, например `BackgroundChecking`, `IndicatorColorIndex` и `InconsistentFormula`. Эти параметры соответствуют параметрам на вкладке Проверка ошибок (Error Checking) в диалоговом окне Сервис⇨Параметры (Tools⇨Options)
Excel4IntlMacroSheets	Sheets	Только чтение. Возвращает коллекцию листов, содержащих международные макросы Excel 4
Excel4MacroSheets	Sheets	Только чтение. Возвращает коллекцию листов, содержащих макросы Excel 4
ExtendList	Boolean	Устанавливает/возвращает необходимость автоматического добавления формул и форматирования при добавлении строк или столбцов в существующие списки строк и столбцов
FeatureInstall	MsoFeatureInstall	Устанавливает/возвращает реакцию Excel на доступ к не установленной функции (это касается как программного, так и интерактивного доступа)
FileConverters	Variant	Только чтение. Параметры: `[Index1]`, `[Index2]`. Возвращает массив всех доступных в Excel преобразователей файлов
FileDialog	FileDialog	Параметры: `[fileDialogType]`. Возвращает объект, который представляет экземпляр одного из нескольких типов диалоговых окон
FileFind	IFind	Возвращает объект, обеспечивающий поиск файлов. Используется в Microsoft Office Macintosh Edition

Имя	*Тип возвращаемого значения*	*Описание*
FileSearch	FileSearch	Только чтение. Возвращает объект, обеспечивающий поиск файлов
FindFormat	FindFormat	Устанавливает/возвращает критерий поиска для типов форматов ячеек, который используется вместе с методами Find и Replace
FixedDecimal	Boolean	Устанавливает/возвращает необходимость определять десятичную точку во вводимых числах с помощью свойства FixedDecimalPlaces
FixedDecimal-Places	Long	Устанавливает/возвращает десятичную точку для всех вводимых в будущем чисел
GenerateGet-PivotData	Boolean	Устанавливает/возвращает возможность получения в Excel данных отчета сводной таблицы
Height	Double	Устанавливает/возвращает высоту основного окна приложения Excel. Это значение невозможно изменить, если основное окно развернуто во весь экран или свернуто в пиктограмму
Hinstance	Long	Только чтение. Возвращает дескриптор экземпляра приложения, вызывающего Excel. В основном используется другими приложениями, например, написанными на Visual Basic
Hwnd	Long	Только чтение. Возвращает дескриптор окна верхнего уровня для окна Excel. В основном используется другими приложениями, например, написанными на Visual Basic
IgnoreRemote-Requests	Boolean	Устанавливает/возвращает возможность игнорировать удаленные запросы DDE
Interactive	Boolean	Устанавливает/возвращает способность Excel реагировать на ввод с клавиатуры или мыши
International	Variant	Только чтение. Параметры: [Index]. Возвращает параметры интернационализации Excel. В качестве значения параметра Index можно использовать константы XlApplicationInternational
Iteration	Boolean	Устанавливает/возвращает необходимость перебора и пересчета всех ячеек в циклических ссылках в попытке определить циклическую ссылку. Свойство используется одновременно со свойствами MaxIterations и MaxChange
LanguageSettings	LanguageSettings	Только чтение. Возвращает объект, описывающий языковые параметры Excel
Left	Double	Устанавливает/возвращает положение левого края основного окна приложения Excel. Это значение не меняется, если окно развернуто во весь экран или свернуто в пиктограмму
LibraryPath	String	Только чтение. Возвращает имя каталога, в котором хранятся надстройки

Имя	*Тип возвращаемого значения*	*Описание*
`MailSession`	`Variant`	Только чтение. Возвращает шестнадцатеричный номер почтового сеанса или значение `Null`, если в данный момент активного сеанса не существует
`MailSystem`	`XlMailSystem`	Только чтение. Возвращает тип используемой на компьютере почтовой системы (например, `xlMapi`, `xlPowerTalk`)
`MapPaperSize`	`Boolean`	Устанавливает/возвращает необходимость автоматической модификации документов, отформатированных для стандартного размера бумаги другой страны/региона (например, для размера A4), чтобы документы правильно печатались на листе стандартного размера в данной местности (например, Letter)
`MathCoprocessor-Available`	`Boolean`	Только чтение. Возвращает информацию о доступности математического сопроцессора
`MaxChange`	`Double`	Устанавливает/возвращает минимальное изменение между итерациями циклической ссылки перед остановкой итерации
`MaxIterations`	`Long`	Устанавливает/возвращает максимальное количество итераций для циклической ссылки
`MemoryFree`	`Long`	Только чтение. Возвращает количество свободной памяти (в байтах), доступной Excel
`MemoryTotal`	`Long`	Только чтение. Возвращает общее количество памяти (в байтах), доступной Excel (включая уже используемую память)
`MemoryUsed`	`Long`	Только чтение. Возвращает количество памяти (в байтах), которая используется Excel
`MouseAvailable`	`Boolean`	Только чтение. Возвращает доступность манипулятора типа мышь
`MoveAfterReturn`	`Boolean`	Устанавливает/возвращает необходимость перемещения текущей ячейки при нажатии клавиши <Enter>
`MoveAfterReturn-Direction`	`XlDirection`	Устанавливает/возвращает направление перемещения курсора при нажатии клавиши <Enter> для смены текущей ячейки
`Name`	`String`	Только чтение. Возвращает "Microsoft Excel"
`Names`	`Names`	Только чтение. Возвращает коллекцию определенных в активной книге имен
`Network-TemplatesPath`	`String`	Только чтение. Возвращает расположение сетевого ресурса, на котором хранятся шаблоны Excel
`NewWorkbookStart`	`Working`	Только чтение. Возвращает объект `StartWorking`
`ODBCErrors`	`ODBCErrors`	Только чтение. Возвращает коллекцию ошибок, полученных в результате последнего запроса или использования отчета сводной таблицы с подключением ODBC

Имя	*Тип возвращаемого значения*	*Описание*
ODBCTimeout	Long	Устанавливает/возвращает время поддержания подключения ODBC (в секундах)
OLEDBErrors	OLEDBErrors	Только чтение. Возвращает коллекцию ошибок, полученных в результате последнего запроса или использования отчета сводной таблицы с подключением OLE DB
OnWindow	String	Устанавливает/возвращает процедуру, которая вызывается каждый раз при активизации окна конечным пользователем
OperatingSystem	String	Только чтение. Возвращает название и версию операционной системы
Organization-Name	String	Только чтение. Возвращает имя организации, отображаемое в диалоговом окне О программе Microsoft Excel
Path	String	Только чтение. Возвращает путь к установочному каталогу Excel
PathSeparator	String	Только чтение. Возвращает обратную косую черту ("\") на ПК и двоеточие ":" на Macintosh
PivotTable-Selection	Boolean	Устанавливает/возвращает использование структурированного выделения в сводной таблице. Например, при выделении заголовка поля выделяются связанные с ним данные
PreviousSelec-tions	Variant	Только чтение. Параметры: [Index]. Возвращает массив последних четырех диапазонов или именованных областей, выделенных с помощью диалогового окна Имя (Name) или диалогового окна Переход (Goto)
ProductCode	String	Только чтение. Возвращает глобально уникальный идентификатор приложения Excel
PromptFor-SummaryInfo	Boolean	Устанавливает/возвращает необходимость запроса у пользователя итоговой информации при попытке сохранения файла
Range	Range	Только чтение. Параметры: Cell1, [Cell2]. Возвращает объект Range, содержащий ячейки, определяемые параметрами
Ready	Boolean	Только чтение. Определяет готовность приложения Excel
RecentFiles	RecentFiles	Только чтение. Возвращает коллекцию последних открытых файлов
RecordRelative	Boolean	Только чтение. Возвращает необходимость использования относительных (True) или абсолютных (False) ссылок на ячейки при записи макросов
ReferenceStyle	XlReference-Style	Устанавливает/возвращает формат ссылок на ячейки: БукваЧисло (например, A1, A3) или НомерСтрокиНомерСтолбца (например, R1C1, R3C1)

Имя	*Тип возвращаемого значения*	*Описание*
RegisteredFunctions	Variant	Только чтение. Параметры: [Index1], [Index2]. Возвращает массив и описание функций, относящихся к внешним библиотекам DLL или ресурсам кода
ReplaceFormat	ReplaceFormat	Устанавливает/возвращает критерий замены для типов форматов ячеек при замене с помощью метода Replace
RollZoom	Boolean	Устанавливает/возвращает необходимость прокрутки или масштабирования при использовании колеса прокрутки на мыши
Rows	Range	Только чтение. Возвращает все строки активного листа
RTD	RTD	Только чтение. Возвращает ссылку на объект данных реального времени (RTD), подключенный к серверу данных реального времени
ScreenUpdating	Boolean	Устанавливает/возвращает необходимость обновления экрана в процессе работы процедуры. Это свойство используется для ускорения работы кода через отключение обновления экрана (для этого свойство необходимо установить в значение False) на время выполнения процедуры. Для обновления экрана вручную можно воспользоваться методом ScreenRefresh
Selection	Object	Только чтение. Возвращает объект, выделенный в данный момент (например, лист, диаграмму)
Sheets	Sheets	Только чтение. Возвращает коллекцию листов активной книги
SheetsInNewWorkbook	Long	Устанавливает/возвращает количество пустых листов, создаваемых в новой книге
ShowChartTipNames	Boolean	Устанавливает/возвращает необходимость отображения подсказок для названий точек данных на диаграмме
ShowChartTipValues	Boolean	Устанавливает/возвращает необходимость отображения подсказок для значений точек данных на диаграмме
ShowStartupDialog	Boolean	Устанавливает/возвращает необходимость отображения панели задач Новая книга (New Workbook) при загрузке приложения Excel
ShowToolTips	Boolean	Устанавливает/возвращает необходимость отображения экранных подсказок в Excel
ShowWindowsInTaskbar	Boolean	Устанавливает/возвращает необходимость отображения каждой книги в панели задач (True) или отображения одного элемента для Excel (False)
SmartTagRecognizers	SmartTagRecognizers	Только чтение. Возвращает коллекцию механизмов распознавания смарт-тегов, которые используются в приложении в данный момент

Имя	*Тип возвращаемого значения*	*Описание*
Speech	Speech	Только чтение. Разрешает доступ к свойствам и методам программного управления голосовых инструментов Office
SpellingOptions	SpellingOptions	Только чтение. Разрешает доступ к орфографическим параметрам приложения
StandardFont	String	Устанавливает/возвращает шрифт, который используется в качестве стандартного шрифта Excel
StandardFontSize	Double	Устанавливает/возвращает размер шрифта, который используется в качестве стандартного размера шрифта в Excel (в пунктах)
StartupPath	String	Только чтение. Возвращает папку, которая используется в качестве стартовой папки Excel
StatusBar	Variant	Устанавливает/возвращает текст строки состояния. Возвращает значение False, если программа Excel перехватила управление строкой состояния. Устанавливается в значение False для передачи Excel управления строкой состояния
TemplatesPath	String	Только чтение. Возвращает путь к шаблонам Excel
ThisCell	Range	Устанавливает/возвращает ячейку, в которой вызывается определенная пользователем функция
ThisWorkbook	Workbook	Только чтение. Возвращает книгу, содержащую работающий в данный момент код VBA
ThousandsSeparator	String	Устанавливает/возвращает символ, который используется в качестве разделителя тысяч. Это глобальный параметр, затрагивающий все открытые книги. Для глобального сброса собственных разделителей воспользуйтесь оператором ApplicationUseSystemSeparators = True
Top	Double	Устанавливает/возвращает положение верхнего края основного окна приложения Excel. Это значение не может быть установлено, если окно развернуто во весь экран или свернуто в пиктограмму
TransitionMenuKey	String	Устанавливает/возвращает клавишу, которая вызывает меню Excel. По умолчанию используется клавиша </>
TransitionMenuKeyAction	Long	Устанавливает/возвращает реакцию на нажатие клавиши <Transition Menu>. При этом может отображаться меню Excel (xlExcelMenu) или диалоговое окно Справка Lotus (Lotus Help) (xlLotusHelp)
TransitionNavigKeys	Boolean	Устанавливает/возвращает состояние клавиш Transition Navigation. Эти клавиши обеспечивают другой способ перемещения и выделения элементов листа

Имя	*Тип возвращаемого значения*	*Описание*
UsableHeight	Double	Только чтение. Возвращает вертикальное пространство (в пунктах), доступное в окне Excel. Это пространство доступно для окна листа. Если свободное пространство недоступно, свойство возвращает значение 1
UsableWidth	Double	Только чтение. Возвращает горизонтальное пространство (в пунктах), доступное в окне Excel. Это пространство доступно для окна листа. Значение этого свойства будет недействительным, если свободное пространство отсутствует. Для проверки свободного пространства проверяйте значение свойства UsableHeight (>1)
UsedObjects	UsedObjects	Только чтение. Представляет объекты, выделенные в книге
UserControl	Boolean	Только чтение. Возвращает значение True, если текущий сеанс Excel был запущен пользователем, и значение False, если текущий сеанс Excel был запущен программно
UserLibraryPath	String	Только чтение. Возвращает расположение надстроек COM
UserName	String	Устанавливает/возвращает имя пользователя в Excel. Обратите внимание, что это имя отображается на вкладке Общие (General) в окне Параметры (Options) и не совпадает с сетевым идентификатором пользователя или именем, которое отображается на заставке Excel
UseSystem-Separators	Boolean	Устанавливает/возвращает необходимость использования системных разделителей в Excel. При установке этого свойства в значение False можно воспользоваться свойствами Application.Decimal-Separator и Application.Thousands-Separator для переопределения системных разделителей. Системные разделители определяются в диалоговом окне Региональные параметры (Regional Settings) в Панели управления (Control Panel)
Value	String	Только чтение. Возвращает строку “Microsoft Excel”
VBE	VBE	Только чтение. Возвращает объект, позволяющий управлять редактором VBE
Version	String	Только чтение. Возвращает версию Excel
Visible	Boolean	Устанавливает/возвращает видимость Excel для пользователя
Watches	Watches	Только чтение. Возвращает объект Watches, который представляет все диапазоны, отслеживаемые в процессе пересчета листа

Имя	*Тип возвращаемого значения*	*Описание*
`Width`	`Double`	Устанавливает/возвращает ширину основного окна приложения Excel. Значение этого свойства не может быть установлено, если основное окно развернуто во весь экран или свернуто в пиктограмму
`Windows`	`Windows`	Только чтение. Возвращает все окна, открытые в пределах текущего сеанса Excel
`WindowsForPens`	`Boolean`	Только чтение. Свойство позволяет определить, работает ли Excel в среде с поддержкой рукописного ввода (Windows for Pen Computing environment)
`WindowState-XlWindow`	`State`	Устанавливает/возвращает текущее состояние окна (развернуто во весь экран, свернуто в пиктограмму или нормальное состояние)
`Workbooks`	`Workbooks`	Только чтение. Возвращает все открытые книги (не включая надстройки) из текущего сеанса Excel
`Worksheet-Function`	`Worksheet-Function`	Только чтение. Возвращает объект, содержащий все функции листов Excel, которые могут использоваться в VBA
`Worksheets`	`Sheets`	Только чтение. Возвращает все листы в пределах активной книги

Методы объекта Application

Имя	*Тип возвращаемого значения*	*Параметры*	*Описание*
`Activate-Microsoft-App`		`Index As XlMSApplication`	Активизирует приложение, указанное в качестве значения параметра `XlMSApplication`. Открывает приложение, если оно еще не открыто. Работает так же, как функция `GetObject` в VBA
`AddChart-AutoFormat`		`Chart, Name AsString, [Description]`	Добавляет форматирование и легенды объекта Chart, указанного в виде параметра, в собственные типы диаграмм
`AddCustom-List`		`ListArray, [ByRow]`	Добавляет массив строк, определенный параметром `ListArray`, в собственные списки Excel. Параметр `ListArray` может указывать на диапазон ячеек
`Calculate`			Пересчитывает все формулы в открытых книгах, изменившиеся с момента последнего пересчета. Имеет смысл только при ручном пересчете
`Calculate-Full`			Пересчитывает все формулы во всех открытых книгах. Запускает принудительный пересчет всех формул в каждой книге вне зависимости от наличия изменений с момента последнего пересчета

Имя	*Тип возвращаемого значения*	*Параметры*	*Описание*
`Calculate-FullRebuild`			Полностью пересчитывает все открытые книги, включая формулы с зависимостями
`Calculate-FullRebuild`			Принудительно запускает полный пересчет данных и перестраивает зависимости для всех открытых книг. Обратите внимание, что зависимости — это формулы, зависящие от значений других ячеек
`Centimeters-ToPoints`	`Double`	`Centimeters As Double`	Выполняет преобразование значения параметра `Centimeters` в пункты, где 1 см = 28.35 пункта
`CheckAbort`	–	`[KeepAbort]`	Останавливает пересчет в приложении Excel
`Check-Spelling`	`Boolean`	`Word As String, [Custom-Dictionary], [Ignore-Uppercase]`	Проверяет орфографию параметра `Word` и возвращает значение `True`, если ошибок не обнаружено и `False`, если ошибки обнаружены
`Convert-Formula`	`Variant`	`Formula, FromReference Style As XlReference Style, [ToReference-Style], [ToAbsolute], [RelativeTo]`	Выполняет преобразование значения параметра `Formula` из формата ссылок R1C1 в формат ссылок A1 и наоборот. Возвращает преобразованную формулу. Кроме этого, позволяет преобразовывать значение параметра `Formula` в форматы с абсолютными и относительными ссылками. Для этого используется параметр `ToReferenceStyle` и константы `XlReferenceStyle`
`DDEExecute`		`Channel As Long, String As String`	Отправляет приложению объект `Command` с помощью механизма DDE через канал, номер которого указывается в качестве значения параметра `Command`. Свойства, названия которых начинаются с DDE, связаны с устаревшей технологией Dynamic Data Exchange. Эта технология использовалась для совместного доступа приложений к данным
`DDEInitiate`	`Long`	`App As String, Topic As String`	Возвращает номер канала, который должен использоваться для обмена данными по технологии DDE для указанного имени приложения и раздела DDE
`DDEPoke`		`Channel As Long,Item, Data`	Отправляет значение параметра `Data` элементу приложения через указанный канал DDE

Имя	*Тип возвращаемого значения*	*Параметры*	*Описание*
`DDERequest`	`Variant`	`Channel As Long, Item As String`	Возвращает информацию для указанного канала и элемента DDE
`DDETerminate`		`Channel As Long`	Закрывает указанный канал DDE
`DeleteChartAutoFormat`		`Name As String`	Удаляет указанный в параметре `Name` собственный тип диаграммы
`DeleteCustomList`		`ListNum As Long`	Удаляет собственный список с указанным номером. Первые четыре списка являются встроенными и не могут быть удалены
`DoubleClick`			Запускается двойным щелчком на активной ячейке активного листа
`Evaluate`	`Variant`	`Name`	Оценивает строковое выражение из параметра `Name`, как при вводе в ячейку листа
`ExecuteExcel4Macro`	`Variant`	`String As String`	Выполняет макрос Excel 4, указанный в параметре `String`. Результат выполнения макроса передается в виде возвращаемого значения метода
`FindFile`	`Boolean`		Отображает диалоговое окно Открыть (Open), позволяя пользователю выбрать файл. При удачном открытии файла возвращается значение `True`
`GetCustomListContents`	`Variant`	`ListNum As Long`	Возвращает собственный список, на который указывает параметр `ListNum`. Список возвращается в виде массива строк
`GetCustomListNum`	`Long`	`ListArray`	Возвращает номер списка, совпадающего с указанным массивом строк. Если ни один из списков не соответствует массиву строк, возвращается значение 0
`GetOpenFilename`	`Variant`	`[FileFilter], [FilterIndex], [Title], [ButtonText], [MultiSelect]`	Отображает диалоговое окно Открыть (Open) с необязательными фильтрами файлов, заголовками и текстами кнопок, которые передаются в качестве параметров метода. Имя и путь к файлу передаются в качестве возвращаемого значения. При установке параметра `MultiSelect` в значение True метод возвращает массив имен файлов. Открытие файла методом не выполняется

Имя	*Тип возвращаемого значения*	*Параметры*	*Описание*
GetPhonetic	String	[Text]	Возвращает фонетический текст японских символов, переданных в качестве значения параметра Text. Если параметр Text не предоставляется, возвращается альтернативный фонетический текст предыдущего значения параметра Text
GetSaveAs-Filename	Variant	[Initial-Filename], [FileFilter], [FilterIndex], [Title], [ButtonText]	Отображает диалоговое окно Сохранить Как (Save As) с принятым по умолчанию необязательным именем файла, фильтрами файлов, заголовками и текстами кнопок, которые передаются в качестве значений параметров. Путь и имя файла передаются в качестве возвращаемого значения метода. Сохранение файла методом не выполняется
Goto		[Reference], [Scroll]	Выделяет объект, на который указывает параметр Reference. При этом активизируется лист, содержащий объект. Параметр Reference может указывать на ячейку, диапазон или имя процедуры VBA. Если параметр Scroll установлен в значение True, выделенный объект будет размещен в верхнем левом углу окна Excel
Help		[HelpFile], [Help-ContextID]	Отображает раздел справочного руководства, на который указывает параметр HelpContextID. Файл справочного руководства указывается в параметре HelpFile
InchesTo-Points	Double	Inches As Double	Выполняет преобразование параметра Inches в пункты и возвращает новое значение (1 дюйм равен 72 пунктам)
InputBox	Variant	Prompt As String, [Title], [Default], [Left], [Top], [HelpFile], [HelpContext-ID], [Type]	Выводит простое окно ввода, похожее на окно ввода, предоставляемое стандартной функцией VBA. При этом параметр Type может использоваться для установки типа возвращаемого значения, например, формула (0), число (1), текст (2), Boolean (4), ссылка на ячейку (8), ошибочное значение (16) или массив значений (64)
Intersect	Range	Arg1 As Range, Arg2 As Range, [Arg3],... [Arg30]	Возвращает пересечение диапазонов, указанных в виде значений параметров. Возвращаемое значение имеет тип Range

Имя	*Тип возвращаемого значения*	*Параметры*	*Описание*
MacroOptions		[Macro], [Description], [HasMenu], [MenuText], [HasShortcut Key], [ShortcutKey], [Category], [StatusBar], [Help-ContextID], [HelpFile]	Обеспечивает модификацию атрибутов макроса, например имени, описания, клавиши быстрого запуска, категории и связанного файла справочного руководства. Является эквивалентом диалогового окна Параметры макроса
MailLogoff			Завершает текущий почтовый сеанс MAPI (например, Exchange, Outlook)
MailLogon		[Name], [Password], [Download-NewMail]	Подключается к принятому по умолчанию клиенту электронной почты MAPI (например, Exchange, Outlook). Принимает такие параметры, как имя и пароль
NextLetter		Workbook	Используется в системах Macintosh вместе с расширениями PowerTalk для открытия самой старой непрочитанной книги из лотка Входящие (In Tray). В операционной системе Windows генерирует сообщение об ошибке
OnKey		Key As String, [Procedure]	Вызывает процедуру, на которую указывает параметр `Procedure`. Процедура запускается при каждом нажатии клавиши или комбинации клавиш, которые указываются в параметре `Key`
OnRepeat		Text As String, Procedure As String	Указывает текст, который выводится в пункте меню Правка⇨Повтор (Edit⇨Repeat), а также процедуру, вызываемую при выборе команды Правка⇨Повтор (Edit⇨Repeat)
OnTime		EarliestTime, Procedure As String, [LatestTime], [Schedule]	Указывает процедуру, которая запускается в момент, определенный параметром `EarliestTime`. Параметр `LatestTime` позволяет описать диапазон времени
OnUndo		Text As String, Procedure As String	Указывает текст, который выводится в пункте меню Правка⇨Отменить (Edit⇨Undo), а также процедуру, вызываемую при выборе пункта меню Правка⇨Отменить (Edit⇨Undo)
Quit			Завершает работу Microsoft Excel
RecordMacro		[BasicCode], [XlmCode]	Добавляет код из параметра `BasicCode` в записываемый в данный момент макрос

Имя	Тип возвращаемого значения	Параметры	Описание
RegisterXLL	Boolean	Filename As String	Загружает ресурс с кодом, указанный в параметре Filename. При этом выполняется регистрация всех функций и процедур, доступных в этом ресурсе
Repeat			Повторяет последнее действие пользователя. Должен находиться в самом начале процедуры
Run	Variant	[Macro], [Arg1], [Arg2], ... [Arg30]	Запускает макрос или процедуру, указанные в параметре Macro. Этот метод позволяет запускать макросы Excel 4.0
SaveWorkspace		[Filename]	Сохраняет рабочую среду в файле, на который указывает параметр Filename
SendKeys		Keys, [Wait]	Отправляет нажатия клавиш из параметра Keys пользовательскому интерфейсу Microsoft Excel
SetDefaultChart		[FormatName], [Gallery]	Устанавливает принятый по умолчанию тип диаграммы, который используется при программном добавлении диаграмм. Значение параметра FormatName может соответствовать одному из встроенных типов или собственному типу диаграммы
Undo			Отменяет последнее действие, выполненное через пользовательский интерфейс
Union	Range	Arg1 As Range, Arg2 As Range, [Arg3], ... [Arg30]	Возвращает объединение диапазонов, указанных в качестве значений параметров
Volatile		[Volatile]	Установка параметра Volatile в значение True приводит к пересчету значения функции, даже если параметры функции не изменились
Wait	Boolean	Time	Приостанавливает работу макроса и Excel до достижения момента, указанного в значении параметра Time

События объекта Application

Имя	Параметры	Описание
NewWorkbook	Wb As Workbook	Возникает при создании новой книги, которая передается в качестве параметра процедуры обработки события
SheetActivate	Sh As Object	Возникает при активизации листа (лист отображается поверх других листов). Активизированный лист передается в процедуру обработки события в качестве параметра

Имя	*Параметры*	*Описание*
SheetBefore-DoubleClick	Sh As Object, Target As Range, Cancel AsBoolean	Возникает перед двойным щелчком на листе. Лист и потенциальное место двойного щелчка передаются в процедуру обработки события в качестве параметра. Действие двойного щелчка можно отменить. Для этого параметр Cancel необходимо установить в значение True
SheetBefore-RightClick	Sh As Object, Target As Range, Cancel As Boolean	Возникает перед щелчком правой кнопкой мыши на листе. Лист и потенциальное место щелчка правой кнопкой мыши передаются в процедуру обработки события в качестве параметров. Действие щелчка правой кнопкой мыши можно отменить. Для этого параметр Cancel необходимо установить в значение True
SheetCalculate	Sh As Object	Возникает при пересчете листа. Пересчитываемый лист передается в процедуру обработки события в качестве параметра
SheetChange	Sh As Object, Target As Range	Возникает при изменении диапазона на листе, например, при удалении содержимого диапазона, вводе данных, удалении строк или столбцов, вставке данных и т.д. Не возникает при вставке строк/столбцов
SheetDeactivate	Sh As Object	Возникает, когда лист теряет фокус. Лист передается в процедуру обработки события в качестве параметра
SheetFollow-Hyperlink	Sh As Object, Target As Hyperlink	Возникает при щелчке на гиперссылке внутри листа. Лист и гиперссылка передаются в процедуру обработки события в качестве параметров
SheetPivotTable Update	ByVal Sh As Object, Target As PivotTable	Возникает при обновлении отчета сводной таблицы. Лист и отчет сводной таблицы передаются в процедуру обработки события в качестве параметров
SheetSelection-Change	Sh As Object,Target As Range	Возникает при выделении новой ячейки на листе. Новый диапазон и лист передаются в процедуру обработки события в качестве параметров
WindowActivate	Wb As Workbook, Wn As Window	Возникает при активизации окна книги (при отображении окна поверх остальных окон книг). Книга и окно передаются в процедуру обработки события в качестве параметров
Window-Deactivate	Wb As Workbook, Wn As Window	Возникает при потере фокуса окном книги. Книга и окно передаются в процедуру обработки события в качестве параметров
WindowResize	Wb As Workbook, Wn As Window	Возникает при изменении размеров окна книги. Книга и окно передаются в процедуру обработки события в качестве параметров. Событие не возникает при изменении размеров окна приложения Excel
Workbook-Activate	Wb As Workbook	Возникает при активизации книги (книга отображается поверх всех остальных окон). Книга передается в процедуру обработки события в качестве параметра

Имя	*Параметры*	*Описание*
WorkbookAddin-Install	Wb As Workbook	Возникает при добавлении в Excel надстройки, которая одновременно является книгой. Книга надстройки передается в процедуру обработки события в качестве параметра
WorkbookAddin-Uninstall	Wb As Workbook	Возникает при удалении из Excel надстройки, которая одновременно является книгой. Книга надстройки передается в процедуру обработки события в качестве параметра
WorkbookBefore-Close	Wb As Workbook, Cancel As Boolean	Возникает перед закрытием книги. Книга передается в процедуру обработки события в качестве параметра. Закрытие книги можно отменить. Для этого параметр Cancel необходимо установить в значение True
WorkbookBefore-Print	Wb As Workbook, Cancel As Boolean	Возникает перед отправкой книги в печать. Книга передается в процедуру обработки события в качестве параметра. Печать можно отменить. Для этого параметр Cancel необходимо установить в значение True
WorkbookBefore-Save	Wb As Workbook, SaveAsUI As Boolean, Cancel As Boolean	Возникает перед сохранением книги. Книга передается в процедуру обработки события в качестве параметра. Сохранение книги можно отменить. Для этого параметр Cancel необходимо установить в значение True. Если параметр SaveAsUI установлен в значение True, отображается диалоговое окно Сохранить Как (Save As)
Workbook-Deactivate	Wb As Workbook	Возникает при потере книгой фокуса. Книга и окно передаются в процедуру обработки события в качестве параметров
WorkbookNew-Sheet	Wb As Workbook, Sh As Object	Возникает при добавлении нового листа в книгу. Книга и новый лист передаются в процедуру обработки события в качестве параметров
WorkbookOpen	Wb As Workbook	Возникает при открытии книги. Открытая книга передается в процедуру обработки события в качестве параметра
WorkbookPivot-TableClose-Connection	ByVal Wb As Workbook, Target As PivotTable	Возникает при разрыве подключения к отчету сводной таблицы. Выделенная книга и отчет сводной таблицы передаются в процедуру обработки события в качестве параметров
WorkbookPivot-TableOpen-Connection	ByVal Wb As Workbook, Target As PivotTable	Возникает при создании подключения к отчету сводной таблицы. Выделенная книга и отчет сводной таблицы передаются в процедуру обработки события в качестве параметров

Пример: объект Application

В этом примере показано, как использовать метод Application.GetOpenFileName для получения имени файла, который необходимо открыть. Ключевой особенностью применения этой функции является присвоение возвращаемого значения переменной типа Variant.

```
Sub UsingGetOpenFilename()
   Dim sFilter As String
   Dim vaFile As Variant
   ' Создать список фильтров. Если опустить пробел перед первой запятой,
   ' Excel не покажет шаблон (*.New).
   sFilter = "New Files (*.New) ,*.new," & _
             "Old Files (*.Old) ,*.old," & _
             "All Files (*.*) ,*.*"
   ' Показать диалоговое окно Открыть файл (File Open).
   ' Возвращаемое значение имеет тип Variant.
   vaFile = Application.GetOpenFilename(FileFilter:=sFilter, _
FilterIndex:=1, _
                              Title:="Открыть новый или старый файл",
                              MultiSelect:=False)
   ' Щелкнул ли пользователь на кнопке Отмена (Cancel)?
   If vaFile <> False Then
      MsgBox "Хотите открыть файл " & vaFile
   End If
End Sub
' Объект Application используется для хранения и получения собственных
' порядков сортировки:
Sub UsingACustomSortOrder()
   Dim vaSortList As Variant
   Dim iListNum As Integer
   Dim bAdded As Boolean
   ' Отсортировать продукты в указанном порядке
   vaSortList = Array("Oranges", "Mangoes", "Apples", "Pears")
   ' Получить номер порядка сортировки, если он существует.
   iListNum = Application.GetCustomListNum(vaSortList)
   ' Если его не существует, возвращается 0, а не сообщение об ошибке
   If iListNum = 0 Then
      ' Создать собственный список для этого порядка сортировки
      Application.AddCustomList vaSortList

      ' Получить его номер (последний!)
      iListNum = Application.CustomListCount

      ' Не забудьте, что он был добавлен и его необходимо удалить.
      bAdded = True
   End If
   ' Отсортировать диапазон в соответствии со списком сортировки.
   ' Обратите внимание, что к номеру списка необходимо добавить 1,
   ' так как 'ordercustom:=1' означает использование стандартного
   ' порядка сортировки (а не собственного списка сортировки)
   ActiveCell.CurrentRegion.Sort key1:=ActiveCell, _
                              ordercustom:=iListNum + 1, header:=xlYes
   ' Необходимо удалить добавленный список.
   If bAdded Then Application.DeleteCustomList iListNum
End Sub
```

Коллекция Areas

В коллекции `Areas` хранятся коллекции объектов `Range`. Каждый объект `Range` представляет блок ячеек (например, A1:A10) или единственную ячейку. Коллекция `Areas` может хранить несколько диапазонов из разных частей книги. Родительским объектом для коллекции `Areas` является объект `Range`.

Общие свойства коллекции Areas

Определения свойств `Application`, `Creator` и `Parent` приводились в начале приложения.

Свойства объекта Areas

Имя	*Тип возвращаемого значения*	*Описание*
`Count`	`Long`	Только чтение. Возвращает количество объектов `Range`, которые содержатся в области
`Item`	`Range`	Параметры: `Index As Long`. Возвращает один объект `Range` из коллекции `Areas`. Параметр `Index` соответствует порядку выделенных диапазонов

Пример: коллекция Areas

Если при использовании объекта `Range`, содержащего несколько различных областей, двадцатая ячейка не входит в первую область диапазона, код вида `rgRange.Cells(20).Value` применять нельзя. Это связано с тем, что Excel выполняет поиск только в первой области, неявно используя вызов `rgRange.Areas(1).Cells(20).Value`. В этом примере показана функция для обхода этой проблемы:

```
Sub TestMultiAreaCells()
    Dim oRNg As Range
    ' Определить диапазон, состоящий из нескольких областей
    Set oRNg = Range("D2:F5,H2:I5")
    ' Ячейка F5 должна быть двенадцатой.
    MsgBox "Rng.Cells(12) is " & oRNg.Cells(12).Address & _
            vbCrLf & "Rng.Areas(1).Cells(12) is " & _
                        oRNg.Areas(1).Cells(12).Address & _
            vbCrLf & "MultiAreaCells(Rng, 12) is " & _
                        MultiAreaCells(Rng, 12).Address

    ' Ячейка H2 должна быть 13 ячейкой составного диапазона и
    ' первой ячейкой во второй области.
    MsgBox "Rng.Cells(13) is " & oRNg.Cells(13).Address & _
            vbCrLf & "Rng.Areas(1).Cells(13) is " & _
                        oRNg.Areas(1).Cells(13).Address & _
            vbCrLf & "MultiAreaCells(Rng, 13) is " & _
                        MultiAreaCells(Rng, 13).Address
End Sub
Function MultiAreaCells(oRange As Range, iCellNum As Long) As Range
    Dim iTotCells As Long, oArea As Range
    ' Перебрать все области диапазона,
    ' начиная с первой, если области закончились.
    Do
        For Each oArea In oRange.Areas
            ' Входит ли интересующая ячейка в эту область?
            If iTotCells + oArea.Cells.Count >= iCellNum Then

                ' Да - вернуться и завершить работу функции
                Set MultiAreaCells = oArea.Cells(iCellNum -f iTotCells)
                Exit Function
            Else
                ' Нет - отсчитать ячейки и продолжить
                iTotCells = iTotCells + oArea.Cells.Count
            End If
        Next
    Loop
End Function
```

Объект AutoCorrect

Объект AutoCorrect обеспечивает доступ ко всей функциональности автозамены (AutoCorrect) в Excel.

Общие свойства объекта AutoCorrect

Определения свойств Application, Creator и Parent приводились в начале приложения.

Свойства объекта AutoCorrect

Имя	*Тип возвращаемого значения*	*Описание*
AutoExpand-ListRange	Boolean	Устанавливается в значение True для автоматического развертывания в списках
Capitalize-NamesOfDays	Boolean	Устанавливает/возвращает необходимость преобразования в верхний регистр первых букв названий дней недели
CorrectCaps-Lock	Boolean	Устанавливает/возвращает необходимость автоматического исправления ошибок набора, возникающих при включении клавиши <Caps Lock>
Correct-SentenceCap	Boolean	Устанавливает/возвращает необходимость преобразования в верхний регистр первой буквы предложения
DisplayAuto-CorrectOptions	Boolean	Отображает/скрывает кнопку Параметры автозамены (AutoCorrect Options). По умолчанию свойство установлено в значение True. Значение свойства влияет на весь пакет Office. Изменение значения свойства в Excel повлияет на все остальные приложения Excel
ReplaceText	Boolean	Устанавливает/возвращает необходимость автоматической замены некоторых слов на слова из списка автозамены
TwoInitial-Capitals	Boolean	Устанавливает/возвращает необходимость автоматического преобразования второй буквы слова в нижний регистр, если первая буква находится в верхнем регистре

Методы объекта AutoCorrect

Имя	*Тип возвращаемого значения*	*Параметры*	*Описание*
Add-Replacement	Variant	What As String, Replacement As String	Добавляет слово (значение параметра What), которое будет автоматически заменяться на другое слово (значение параметра Replacement) в массив списка замены (ReplacementList)
Delete-Replacement	Variant	What As String	Удаляет слово из списка ReplacementList. При этом автоматическая замена слова на другое слово отключается

Имя	Тип возвращаемого значения	Параметры	Описание
ReplacementList	Variant	[Index]	Возвращает многомерный массив строк. Первый столбец массива содержит заменяемое слово, а во втором столбце хранится замещающий текст. Параметр Index можно использовать для получения массива, содержащего слово и его замену

Пример: объект AutoCorrect

В этом примере объект AutoCorrect используется для поиска замены конкретного слова.

```
Sub TestAutoCorrect()
   MsgBox "'(c)' заменяется на " & UseAutoCorrect("(c)")
End Sub
Function UseAutoCorrect(ByVal sWord As String) As String
   Dim i As Integer
   Dim vaRepList As Variant
   Dim sReturn As String
   ' По умолчанию возвращать исходное слово
   sReturn = sWord
   ' Загрузить список замен в массив
   vaRepList = Application.AutoCorrect.ReplacementList
   ' Перебрать список замен
   For i = LBound(vaRepList) To UBound(vaRepList)
      ' Найдено ли совпадение?
      If vaRepList(i, 1) = sWord Then
         ' Вернуть текст замены
         sReturn = vaRepList(i, 2)
         ' Выйти из цикла
         Exit For
      End If
   Next
   ' Вернуть слово или его замену, если такая замена существует.
   UseAutoCorrect = sReturn
End Function
```

Объект AutoFilter

Объект AutoFilter предоставляет доступ к функциональности автофильтра в Excel. Объект поддерживает программную фильтрацию диапазона с текстом для получения строк конкретных типов, скрывая строки, которые не соответствуют определенному критерию. Примеры фильтров поддерживают получение первых десяти строк столбца, получение строк, равных определенному значению, а также получение непустых ячеек в строке. Для доступа к этим возможностям в Excel можно воспользоваться командой меню Данные⇨Фильтр⇨Автофильтр (Data⇨Filter⇨AutoFilter). Для объекта AutoFilter родительским является объект Worksheet (подразумевается, что в пределах листа может присутствовать только один автофильтр).

Объект AutoFilter используется совместно с методом AutoFilter объекта Range и свойством AutoFilterType объекта Worksheet.

Общие свойства объекта AutoFilter

Определения свойств Application, Creator и Parent приводились в начале приложения.

Свойства объекта AutoFilter

Имя	*Тип возвращаемого значения*	*Описание*
Filters	Filters	Только чтение. Возвращает коллекцию фильтров, связанных с диапазоном (например, непустые строки)
Range	Range	Только чтение. Возвращает группу ячеек, к которой применяется автофильтр

Пример: объект AutoFilter

В этом примере показано использование объектов AutoFilter, Filters и Filter. Приведенный ниже код выводит список всех применяемых в данный момент фильтров:

```
Sub ShowAutoFilterCriteria()
   Dim oAF As AutoFilter, oFlt As Filter
   Dim sField As String
   Dim sCrit1 As String, sCrit2 As String
   Dim sMsg As String, i As Integer
   ' Проверить существование фильтров на листе
   If ActiveSheet.AutoFilterMode = False Then
      MsgBox "Автофильтр на листе отсутствует"
      Exit Sub
   End If
   ' Получить объект автофильтра, установленного на листе
   Set oAF = ActiveSheet.AutoFilter
   ' Перебрать все объекты Filters внутри объекта AutoFilter
   For i = 1 To oAF.Filters.Count
      ' Получить имя поля из первой строки
      ' диапазона с автофильтром
      sField = oAF.Range.Cells(1, i).Value
      ' Получить объект Filter
      Set oFlt = oAF.Filters(i)
      ' Если фильтр включен...
      If oFlt.On Then
         ' Получить стандартный критерий фильтрации
         sMsg = sMsg & vbCrLf & sField & oFlt.Criteria1
         ' Если это специальный фильтр, отобразить его
         Select Case oFlt.Operator
            Case xlAnd
               sMsg = sMsg & " And " & sField & oFlt.Criteria2
            Case xlOr
               sMsg = sMsg & " Or " & sField & oFlt.Criteria2
            Case xlBottom10Items
               sMsg = sMsg & " (bottom 10 items)"
            Case xlBottom10Percent
               sMsg = sMsg & " (bottom 10%)"
            Case xlTop10Items
               sMsg = sMsg & " (top 10 items)"
            Case xlTop10Percent
               sMsg = sMsg & " (top 10%)"
         End Select
      End If
   Next
```

```
    If sMsg = "" Then
        ' Фильтры не установлены. Сообщить об этом
        sMsg = "Диапазон " & oAF.Range.Address & " не содержит фильтров."
    Else
        ' Фильтры установлены. Показать фильтры
        sMsg = "Диапазон " & oAF.Range.Address & _
               " содержит следующие фильтры:" & sMsg
    End If
    ' Вывести сообщение
    MsgBox sMsg
End Sub
```

Объект AutoRecover

Этот объект обеспечивает доступ к параметрам автоматического восстановления приложения Excel, которые доступны на вкладке Сохранение (Save) в диалоговом окне Параметры (Options). Изменение этих параметров отражается на всех книгах. Обратите внимание, что каждая книга может выбирать необходимость использования автоматического восстановления. Этот параметр также доступен на вкладке Сохранение (Save).

Общие свойства объекта AutoRecover

Определения свойств `Application`, `Creator` и `Parent` приводились в начале приложения.

Свойства объекта AutoRecover

Имя	*Тип возвращаемого значения*	*Описание*
Enabled	Boolean	Возвращает значение `True`, если объект `AutoRecover` включен
Path	String	Устанавливает/возвращает полный путь к каталогу, в котором хранятся временные файлы для автоматического восстановления
Time	Long	Устанавливает/возвращает интервал времени для объекта `AutoRecover`. Возможные значения свойства находятся в диапазоне от 1 до 120 минут (по умолчанию свойство установлено в значение 10)

Пример: объект AutoRecover

В следующей подпрограмме и в функции устанавливаются свойства объекта `AutoRecover`. После этого они используются в коде соответствующей книги.

```
Sub SetAutoRecoverOptions()
    ' Установить параметры автоматического восстановления для приложения
    ChangeAutoRecoverSettings True, "C:\Documents and
↳Settings\AgamaOffice\My
↳Documents\Backup Files\AutoRecover\Excel", 2

    ' Удостовериться, что установленные параметры используются в книге
    ThisWorkbook.EnableAutoRecover = True
End Sub
Function ChangeAutoRecoverSettings(Optional ByVal vEnable As Variant,
Optional ByVal vPath As Variant, Optional ByVal vTime As Variant)
    With Application.AutoRecover
        ' Устанавливать свойство только при наличии значения параметра
```

```
        If Not IsMissing(vEnable) Then
            ' Включить автоматическое восстановление
            .Enabled = vEnable
        End If

        ' Устанавливать свойство только при наличии значения параметра
        If Not IsMissing(vPath) Then
            ' Изменить путь к центральному хранилищу временных файлов
            .Path = vPath
        End If
        ' Устанавливать свойство только при наличии значения параметра
        If Not IsMissing(vTime) Then
            ' Сохранять файлы для автоматического восстановления
            ' каждые 2 минуты
            .Time = vTime
        End If
    End With
End Function
```

Объект Axis и коллекция Axes

В коллекции Axes хранятся все объекты Axis в пределах диаграммы Excel. Каждый объект Axis является эквивалентом оси на диаграмме Excel (например, ось X, ось Y и т.д.). Родительским объектом для коллекции Axes является объект Chart.

Кроме типичных атрибутов, коллекция Axes предоставляет свойство Count, которое возвращает количество объектов Axis в пределах коллекции. Кроме того, в отличие от большинства других коллекций, метод Item коллекции Axes принимает два параметра: Type и AxisGroup. В качестве значений параметра Type можно использовать одну из констант xlAxisType (xlValue, xlCategory или xlSeriesAxis). Необязательный второй параметр, AxisGroup, может принимать в качестве значения одну из констант xlAxisGroup (xlPrimary или xlSecondary).

Общие свойства объекта Axis

Определения свойств Application, Creator и Parent приводились в начале приложения.

Свойства объекта Axis

Имя	*Тип возвращаемого значения*	*Описание*
AxisBetweenCategories	Boolean	Устанавливает/возвращает пересечение оси значений с осью категорий между категориями (как в гистограмме) или параллельное размещение оси значений с метками категорий (как в диаграмме графика)
AxisGroup	XlAxisGroup	Только чтение. Возвращает группу текущей оси (основная, xlPrimary, или вторичная, xlSecondary)
AxisTitle	AxisTitle	Только чтение. Возвращает объект для управления свойствами заголовков оси
BaseUnit	XlTimeUnit	Устанавливает/возвращает тип базовых единиц измерения для оси категорий. Применяется совместно со свойством BaseUnitIsAuto. Использование совместно с осью значений является ошибкой

Имя	*Тип возвращаемого значения*	*Описание*
BaseUnitIs-Auto	Boolean	Устанавливает/возвращает автоматический выбор единиц измерения для оси категорий. Использование совместно с осью значений является ошибкой
Border	Border	Только чтение. Возвращает свойства границы для выделенной оси
Category-Names	Variant	Устанавливает/возвращает имена категорий для осей в виде строкового массива
Category-Type	XlCategoryType	Устанавливает/возвращает тип оси, которая используется в качестве оси категорий. Применение совместно с осью значений является ошибкой
Crosses	XlAxisCrosses	Устанавливает/возвращает точку пересечения оси с другой осью: минимальное значение, максимальное значение, автоматически выбирается Excel или собственное значение
CrossesAt	Double	Устанавливает/возвращает значение, в котором другие оси пересекают текущую ось. Используется, когда свойство Crosses установлено в значение xlAxisCrossessCustom
DisplayUnit	XlDisplayUnit	Устанавливает/возвращает тип единиц измерения, отображаемых для осей (например, xlThousands)
DisplayUnit-Custom	Double	Устанавливает/возвращает значение отображаемых единиц, если свойство DisplayUnit установлено в значение xlCustom
DisplayUnit-Label	DisplayUnit-Label	Только чтение. Возвращает объект, позволяющий управлять метками единиц измерения оси
HasDisplay-UnitLabel	Boolean	Устанавливает/возвращает необходимость отображения метки единицы измерения, созданной с помощью свойства DisplayUnit или свойства DisplayUnitCustem
HasMajor-Gridlines	Boolean	Устанавливает/возвращает необходимость отображения основных линий на оси
HasMinor-Gridlines	Boolean	Устанавливает/возвращает необходимость отображения дополнительных линий на оси
HasTitle	Boolean	Устанавливает/возвращает необходимость отображения заголовка оси
Height	Double	Только чтение. Возвращает высоту оси
Left	Double	Только чтение. Возвращает положение оси относительно левого края диаграммы
Major-Gridlines	Gridlines	Только чтение. Возвращает объект, позволяющий управлять форматированием основных линий
MajorTick-Mark	XlTickMark	Устанавливает/возвращает внешний вид основных меток (например, внутри оси, за пределами оси)
MajorUnit	Double	Устанавливает/возвращает значение основной единицы измерения
MajorUnit-IsAuto	Boolean	Устанавливает/возвращает необходимость автоматической установки значения свойства MajorUnit

Имя	*Тип возвращаемого значения*	*Описание*
MajorUnit-Scale	XlTimeUnit	Устанавливает/возвращает тип основной единицы измерения
MaximumScale	Double	Устанавливает/возвращает максимальное значение на оси
Maximum-ScaleIsAuto	Boolean	Устанавливает/возвращает необходимость автоматического определения максимального значения на оси
MinimumScale	Double	Устанавливает/возвращает минимальное значение на оси
Minimum-ScaleIsAuto	Boolean	Устанавливает/возвращает необходимость автоматического определения минимального значения на оси
Minor-Gridlines	Gridlines	Только чтение. Возвращает объект, позволяющий управлять форматированием дополнительных направляющих
MinorTick-Mark	XlTickMark	Устанавливает/возвращает внешний вид дополнительных меток (например, внутри оси, за пределами оси)
MinorUnit	Double	Устанавливает/возвращает значение дополнительной единицы измерения
MinorUnit-IsAuto	Boolean	Устанавливает/возвращает необходимость автоматического определения значения свойства MinorUnit
MinorUnit-Scale	XlTimeUnit	Устанавливает/возвращает масштаб для дополнительных единиц измерения
Reverse-PlotOrder	Boolean	Устанавливает/возвращает необходимость обратного отсчета значений на оси
ScaleType	XlScaleType	Устанавливает/возвращает тип масштабирования для единиц измерения: линейное или логарифмическое
TickLabel-Position	XlTickLabel-Position	Устанавливает/возвращает положение меток относительно оси (например, выше, ниже)
TickLabels	TickLabels	Только чтение. Возвращает объект, позволяющий управлять свойствами меток на оси
TickLabel-Spacing	Long	Устанавливает/возвращает частоту отображения меток
TickMark-Spacing	Long	Устанавливает/возвращает частоту отображения меток на оси. Использование вместе с осью значений является ошибкой
Top	Double	Только чтение. Возвращает положение верхнего края оси относительно верхнего края диаграммы
Type	XlAxisType	Устанавливает/возвращает тип оси (xlCategory, xlSeriesAxis или xlValue)
Width	Double	Только чтение. Возвращает ширину оси

Методы объекта Axis

Имя	*Тип возвращаемого значения*	*Параметры*	*Описание*
Delete	Variant		Удаляет ось из коллекции Axes
Select	Variant		Выделят ось на диаграмме

Пример: объект Axis и коллекция Axes

В этом примере устанавливаются метки для оси X (вне зависимости от лежащих в основе диаграммы данных) и применяется соответствующее форматирование.

```
Sub FormatXAxis()
   Dim oCht As Chart, oAxis As Axis
   ' Получить первую встроенную диаграмму на листе
   Set oCht = ActiveSheet.ChartObjects(1).Chart
   ' Получить ось X этой диаграммы
   Set oAxis = oCht.Axes(xlCategory)
   ' Выполнить форматирование оси X
   With oAxis
      .CategoryNames = Array("Item 1", "Item 2", "Item 3")
      .TickLabels.Orientation = 45
      .AxisBetweenCategories = True
      .ReversePlotOrder = False
      .MinorTickMark = xlTickMarkNone
 .MajorTickMark = xlTickMarkCross
   End With
End Sub
```

Объект AxisTitle

В объекте AxisTitle хранится форматирование и текст заголовков каждой оси диаграммы. Родительским для объекта AxisTitle является объект Axis. Объект AxisTitle используется совместно со свойством HasTitle родительского объекта Axis. Для существования дочернего объекта AxisTitle свойство HasTitle должно быть установлено в значение True.

Общие свойства объекта AxisTitle

Определения свойств Application, Creator и Parent приводились в начале приложения.

Свойства объекта AxisTitle

Имя	*Тип возвращаемого значения*	*Описание*
AutoScale-Font	Variant	Устанавливает/возвращает необходимость автоматического изменения размера шрифта в случае изменения размера родительской диаграммы
Border	Border	Только чтение. Возвращает свойства границы вокруг заголовка выделенной оси
Caption	String	Устанавливает/возвращает текст заголовка оси
Characters	Characters	Только чтение. Параметры: [Start], [Length]. Возвращает объект, содержащий все символы заголовка оси. Объект поддерживает управление отдельными символами
Fill	ChartFill-Format	Только чтение. Возвращает объект, содержащий параметры форматирования фона в заголовке оси диаграммы
Font	Font	Только чтение. Возвращает объект, содержащий параметры шрифта заголовка оси диаграммы

Имя	*Тип возвращаемого значения*	*Описание*
Horizontal-Alignment	Variant	Устанавливает/возвращает параметры горизонтального выравнивания заголовка оси диаграммы. В качестве значений свойства используются константы xlAlign
Interior	Interior	Только чтение. Возвращает объект, содержащий параметры форматирования внутренней области в области текста заголовка оси диаграммы (например, цвет внутренней области)
Left	Double	Устанавливает/возвращает расстояние от левого края области текста заголовка оси до левого края диаграммы
Name	String	Только чтение. Возвращает имя объекта заголовка оси
Orientation	Variant	Устанавливает/возвращает угол текста для заголовка оси. Значение измеряется в градусах (от -90 до 90) или может быть равно одной из констант xlOrientation
ReadingOrder	Long	Устанавливает/возвращает направление текста (слева направо или справа налево). Имеет смысл только в соответствующих языках
Shadow	Boolean	Устанавливает/возвращает необходимость отображения заголовка оси с эффектом тени
Text	String	Устанавливает/возвращает текст заголовка оси
Top	Double	Устанавливает/возвращает расстояние от верхнего края текстовой области заголовка оси до верхнего края диаграммы
Vertical-Alignment	Variant	Устанавливает/возвращает состояние горизонтального выравнивания заголовка оси. В качестве значений этого свойства используются константы xlVAlign

Методы объекта AxisTitle

Имя	*Тип возвращаемого значения*	*Параметры*	*Описание*
Delete	Variant		Удаляет заголовок оси
Select	Variant		Выделяет заголовок оси на диаграмме

Пример: объект AxisTitle

В этом примере к оси X добавляется заголовок. Кроме этого, в заголовок добавляется текст и форматирование.

```
Sub FormatXAxisTitle()
   Dim oCht As Chart, oAT As AxisTitle
   ' Получить первую встроенную диаграмму на листе
   Set oCht = ActiveSheet.ChartObjects(1).Chart
   ' Добавить заголовок к оси X
   oCht.Axes(xlCategory).HasTitle = True
   ' Получить заголовок
   Set oAT = oCht.Axes(xlCategory).AxisTitle
   ' Отформатировать заголовок
   With oAT
      .AutoScaleFont = False
```

```
        .Caption = "Заголовок оси X"
        .Font.Bold = True
    End With
End Sub
```

Объект Border и коллекция Borders

Коллекция Borders содержит свойства четырех границ родительского объекта. Родительскими объектами коллекции Borders являются объект Range и объект Style. Коллекция Borders всегда содержит четыре границы. Для доступа к одному из объектов Border внутри коллекции Borders можно воспользоваться константами xlBordersIndex, которые выступают в качестве значений свойства Item коллекции Borders.

Каждый объект Border соответствует границе или границам вокруг родительского объекта. Некоторые объекты поддерживают доступ ко всем четырем границам, как единому целому (например, независимый доступ к левой границе не поддерживается). Следующие объекты могут выступать в качестве родительских для объекта Border (не коллекции Borders): Axis, AxisTitle, ChartArea, ChartObject, ChartTitle, DataLabel, DataTable, DisplayUnitLabel, Downbars, DropLines, ErrorBars, Floor, GridLines, HiLoLines, LeaderLines, Legend, LegendKey, OleObject, PlotArea, Point, Series, SeriesLines, TrendLine, UpBars и Walls. Данные коллекции также могут выступать в качестве родительских для объекта Border: DataLabels, ChartObjects и OleObjects.

Коллекция Borders предоставляет несколько свойств кроме обычных атрибутов коллекции. Эти свойства перечислены в следующей таблице.

Свойства коллекции Borders

Имя	*Тип возвращаемого значения*	*Описание*
Color	Variant	Устанавливает/возвращает цвет для всех четырех границ внутри коллекции. Для выбора цвета используется функция RGB
ColorIndex	Variant	Устанавливает/возвращает цвет для всех четырех границ внутри коллекции. Для выбора цвета используется порядковый номер цвета в текущей цветовой палитре
Count	Long	Только чтение. Возвращает количество объектов Border в коллекции. Всегда возвращает значение 4
LineStyle	Variant	Устанавливает/возвращает стиль линии, которая используется для отображения границы (например, xlDash). Для установки стиля линии применяются константы xlLineStyle
Value	Variant	Устанавливает/возвращает стиль линии, которая используется для отображения границы (например, xlDash). Для установки стиля линии применяются константы xlLineStyle. Совпадает со свойством LineStyle
Weight	Variant	Устанавливает/возвращает толщину границ, входящих в коллекцию (например, xlThin, xlThick). Для установки значения используются константы xlBorderWeight

Общие свойства объекта Border

Определения свойств Application, Creator и Parent приводились в начале приложения.

Свойства объекта Border

Имя	*Тип возвращаемого значения*	*Описание*
Color	Variant	Устанавливает/возвращает цвет границы. Для выбора цвета используется функция RGB
ColorIndex	Variant	Устанавливает/возвращает цвет границы. Для выбора цвета используется порядковый номер цвета в текущей цветовой палитре
LineStyle	Variant	Устанавливает/возвращает стиль линии, которая используется для отображения границы (например, xlDash). Для установки стиля линии применяются константы xlLineStyle
Weight	Variant	Устанавливает/возвращает толщину границ, входящих в коллекцию (например, xlThin, xlThick). Для установки значения используются константы xlBorderWeight

Пример: объект Border и коллекция Borders

Настройка трехмерного эффекта для диапазона:

```
Sub TestFormat3D()
   ' Отформатировать выделенный диапазон, придав утопленый трехмерный вид
   Format3D Selection
End Sub
Sub Format3D(oRange As Range, Optional bSunken As Boolean = True)
   ' Использование диапазона...
   With oRange
       ' Окружить диапазон белой границей
       .BorderAround Weight:=xlMedium, Color:=RGB(255, 255, 255)

       If bSunken Then
           ' Диапазон утоплен, поэтому левую и верхнюю границу необходимо
           ' сделать темно-серыми
           .Borders(xlEdgeLeft).Color = RGB(96, 96, 96)
           .Borders(xlEdgeTop).Color = RGB(96, 96, 96)
       Else
           ' Диапазон выступает, поэтому темно-серым выделяется
           ' правая и нижняя границы
           .Borders(xlEdgeRight).Color = RGB(96, 96, 96)
           .Borders(xlEdgeBottom).Color = RGB(96, 96, 96)
       End If
   End With
End Sub
```

Коллекция CalculatedFields

Дополнительная информация приводится в разделах "Объект PivotField", "Коллекция PivotFields" и "Коллекция CalculatedFields".

Коллекция CalculatedItems

Дополнительная информация приводится в разделах "Объект PivotField", "Коллекция PivotFields" и "Коллекция CalculatedFields".

Объект CalculatedMember и коллекция CalculatedMembers

Коллекция CalculatedMembers содержит все объекты CalculatedMember из указанной сводной таблицы. Каждый объект CalculatedMember соответствует вычислимому полю или вычислимому элементу.

Общие свойства коллекции CalculatedMembers

Определения свойств Application, Creator и Parent приводились в начале приложения.

Свойства коллекции CalculatedMembers

Имя	*Тип возвращаемого значения*	*Описание*
Count	Long	Возвращает количество объектов в коллекции
Item	CalculatedMember	Параметр: Index As Variant. Возвращает единственный объект CalculatedMember из коллекции CalculatedMembers

Общие свойства объекта CalculatedMember

Определения свойств Application, Creator и Parent приводились в начале приложения.

Свойства объекта CalculatedMember

Имя	*Тип возвращаемого значения*	*Описание*
Formula	String	Возвращает формулу объекта CalculatedMember в формате многомерного выражения (MDX)
IsValid	Boolean	Указывает на успешное создание экземпляра указанного объекта CalculatedMember с помощью поставщика OLAP в течение текущего сеанса. Свойство возвращает значение True, даже если сводная таблица не подключена к источнику данных
Name	String	Устанавливает/возвращает имя объекта
SolveOrder	Long	Возвращает значение аргумента многомерного выражения объекта CalculatedMember (по умолчанию свойство возвращает значение 0)
SourceName	String	Возвращает имя объекта, полученное из исходных данных для указанного отчета сводной таблицы
Type	XlCalculatedMemberType	Возвращает тип объекта CalculatedMember

Методы объекта CalculatedMember

Имя	Тип возвращаемого значения	Параметры	Описание
Delete			Удаляет выделенный объект

Пример: коллекция CalculatedMembers и объект CalculatedMember

Данная подпрограмма возвращает информацию о каждом объекте `CalculateMember` из источника данных, который используется для создания сводной таблицы на листе `wksPivotTable`. Подпрограмма выдает сообщение в том случае, если источник данных не является одним из типов OLAP или в сводной таблице отсутствуют объекты `CalculatedMember`.

```
Sub ReturnCalculatedMembers()
    Dim lIcon As Long, lCount As Long
    Dim ptTable As PivotTable
    Dim oCalcMember As CalculatedMember
    Dim oCalcMembers As CalculatedMembers
    Dim sInfo As String
    ' Установить ссылку на сводную таблицу
    Set ptTable = wksPivotTable.PivotTables("WroxSales1")
    On Error Resume Next
        Set oCalcMembers = ptTable.CalculatedMembers
    On Error GoTo 0

    ' Возвращается ли ссылка на вычислимые элементы?
    If Not oCalcMembers Is Nothing Then
        ' Если в пределах сводной таблицы присутствует хотя бы
        ' один вычислимый элемент...
        If oCalcMembers.Count > 0 Then
            ' Инициализировать счетчик
            ' и переменные сообщений
            lCount = 1
            lIcon = vbInformation

            ' Перебрать вычислимые элементы
            ' и сохранить их имена и формулы
            For Each oCalcMember In oCalcMembers
                With oCalcMember
                    sInfo = sInfo & lCount & ") " & .Name & ": " &
↳ .Formula
                    lCount = lCount + 1
                End With
            Next oCalcMember

        Else
            ' Это действительный источник данных OLAP,
            ' но вычислимые элементы отсутствуют
            lIcon = vbExclamation
            sInfo = "Отсутствуют вычислимые элементы."
        End If
    Else
        ' Коллекция oCalcMembers не содержит элементов.
        ' Это не источник данных OLAP
        lIcon = vbCritical
        sInfo = "Невозможно получить вычислимые элементы. Источник данных не
↳ принадлежит типу OLAP."
    End If
```

```
    MsgBox sInfo, lIcon, "Вычислимые элементы"
End Sub
```

Объект CalloutFormat

Объект CalloutFormat соответствует линиям выноски на геометрических формах. Родительским для объекта CalloutFormat является объект Shape.

Общие свойства объекта CalloutFormat

Определения свойств Application, Creator и Parent приводились в начале приложения.

Свойства объекта CalloutFormat

Имя	*Тип возвращаемого значения*	*Описание*
Accent	MsoTriState	Устанавливает/возвращает необходимость использования вертикальной акцентированной полосы для отделения поля выноски от линии
Angle	MsoCallout-AngleType	Устанавливает/возвращает угол линии выноски относительно поля выноски
AutoAttach	MsoTriState	Устанавливает/возвращает необходимость автоматического изменения соединения линии выноски к полю выноски в зависимости от направления линии (слева или справа от поля выноски)
AutoLength	MsoTriState	Только чтение. Возвращает возможность автоматического изменения размера линии выноски при перемещении составного поля выноски
Border	MsoTriState	Устанавливает/возвращает наличие границы вокруг поля выноски
Drop	Single	Только чтение. Возвращает расстояние от поля выноски до точки, на которую указывает линия выноски
DropType	MsoCallout-DropType	Только чтение. Возвращает точку на поле выноски, к которой присоединяется линия выноски
Gap	Single	Устанавливает/возвращает расстояние между концом линии выноски и полем выноски
Length	Single	Только чтение. Возвращает длину первой части линии выноски. Свойство AutoLength должно быть установлено в значение False
Type	MsoCalloutType	Устанавливает/возвращает тип линии, которая используется для выноски

Пример: объект CalloutFormat

В этом примере устанавливается форматирование выносок на листе:

```
Sub FormatAllCallouts()
   Dim oShp As Shape
   Dim oCF As CalloutFormat
   ' Перебрать все геометрические фигуры на листе
```

```
    For Each oShp In ActiveSheet.Shapes
        ' Это выноска?
        If oShp.Type = msoCallout Then
            ' Да - включить автоматическое изменение размера поля
            oShp.TextFrame.AutoSize = True

            ' Получить объект CalloutFormat
            Set oCF = oShp.Callout

            ' Установить форматирование выноски
            With oCF
                .Gap = 0
                .Border = msoFalse
                .Accent = msoTrue
                .Angle = msoCalloutAngle30
                .PresetDrop msoCalloutDropCenter
            End With
        End If
    Next
End Sub
```

Объект CellFormat

Этот объект одновременно представляет свойства FindFormat и ReplaceFormat объекта Application, которые используются методами Find и Replace (соответственно) объекта Range.

Для поиска формата ячеек внутри диапазона с помощью метода Find сначала необходимо установить значение свойства FindFormat. Для замены форматирования ячеек внутри диапазона с помощью метода Replace сначала следует установить значение свойства ReplaceFormat. Любые значения, присвоенные аргументам What или Replacement, будут комбинироваться с помощью операции And. Например, при поиске слова "wrox" и установке свойства FindFormat для поиска полужирного шрифта будут найдены только те ячейки, в которых присутствует полужирный текст "wrox".

При поиске форматирования необходимо убедиться в том, что параметр SearchFormat метода Find установлен в значение True. При замене форматирования в значение True должен быть установлен параметр ReplaceFormat метода Replace.

Если необходимо найти только форматирование, параметр What метода Find не должен иметь значения. Если нужно заменить только форматирование, параметр Replace метода Replace также не должен иметь значения.

При замене одного форматирования другим необходимо явно указать форматирование, которое будет заменяться новым. Например, если при поиске ячеек, содержащих полужирное выделение и выделение красным цветом, форматирование необходимо заменить на выделение синим цветом (без полужирного выделения), компонент полужирного выделения в свойстве ReplaceFormat необходимо явно установить в значение False. В противном случае будет получено выделение синим цветом и полужирным шрифтом.

При поиске или замене различных параметров форматирования (или вообще без применения параметров форматирования) воспользуйтесь методом Clear объекта CellFormat (если была объявлена соответствующая переменная) или методом Clear свойств FindFormat и ReplaceFormat. Установка параметров SearchFormat и ReplaceFormat методов Find и Replace в значение False не приведет к отказу от использования параметров свойств FindFormat и ReplaceFormat.

Общие свойства объекта CellFormat

Определения свойств Application, Creator и Parent приводились в начале приложения.

Свойства объекта CellFormat

Имя	*Тип возвращаемого значения*	*Описание*
AddIndent	Variant	Показывает, выполняется ли автоматическое смещение текста при установке выравнивания текста в равномерное распределение по вертикали или горизонтали
Borders	Borders	Устанавливает/возвращает критерий поиска в зависимости от формата границ ячейки
Font	Font	Устанавливает/возвращает критерий поиска в зависимости от форматирования шрифта внутри ячейки
Formula-Hidden	Variant	Возвращает необходимость сокрытия формулы при защите листа. Возвращает значение Null, если в указанном диапазоне одновременно содержатся ячейки со скрытыми и без скрытых формул
Horizontal-Alignment	Variant	Устанавливает/возвращает горизонтальное выравнивание указанного объекта
IndentLevel	Variant	Устанавливает/возвращает уровень смещения для ячейки или диапазона
Interior	Interior	Устанавливает/возвращает критерий поиска в зависимости от формата внутренней области ячейки
Locked	Variant	Устанавливает/возвращает возможность модификации ячеек внутри диапазона при защите листа. Возвращает значение Null, если блокируется только часть ячеек диапазона
MergeCells	Variant	Возвращает значение True, если диапазон или стиль содержат ячейки, полученные путем слияния других ячеек
NumberFormat	Variant	Устанавливает/возвращает формат чисел, связанный ячейками внутри диапазона. Возвращает значение Null, если ячейки имеют разный формат чисел
Number-FormatLocal	Variant	Устанавливает/возвращает формат чисел, связанный с ячейками внутри диапазона. Формат возвращается на языке пользовательского интерфейса. Возвращает значение Null, если ячейки имеют разный формат чисел
Orientation	Variant	Устанавливает/возвращает ориентацию текста для ячеек. Для установки свойства могут использоваться значения в пределах от -90 до 90 или константы XlOrientation
ShrinkToFit	Variant	Устанавливает/возвращает необходимость автоматического изменения размера текста внутри ячейки в соответствии с изменившимися размерами столбца. Возвращает значение Null, если строки диапазона имеют различные значения свойства ShrinkToFit

Имя	*Тип возвращаемого значения*	*Описание*
`Vertical-Alignment`	`Variant`	Устанавливает/возвращает вертикальное выравнивание ячеек диапазона. Для установки свойства можно воспользоваться константами `XLVAlign`
`WrapText`	`Variant`	Устанавливает/возвращает перенос текста внутри ячейки. Возвращает значение `Null`, если ячейки внутри диапазона используют различные параметры переноса текста

Методы объекта CellFormat

Имя	*Тип возвращаемого значения*	*Параметры*	*Описание*
`Clear`			Удаляет критерии, установленные в свойствах `FindFormat` и `ReplaceFormat`

Пример: объект CellFormat

Следующая подпрограмма выполняет поиск в используемом диапазоне на листе. При этом все ячейки, содержащие шрифт Tahoma и светло-синий фон, заменяются на ячейки со шрифтом Arial и светло-зеленым фоном.

```
Sub ReplaceFormats()

    Dim oCellFindFormat As CellFormat
    Dim oCellReplaceFormat As CellFormat
    Dim rngReplace As Boolean, sMessage As String

    ' Определить переменные для форматов поиска и замены
    Set oCellFindFormat = Application.FindFormat
    Set oCellReplaceFormat = Application.ReplaceFormat

    ' Установить критерий поиска
    With oCellFindFormat
        .Clear
        .Font.Name = "Tahoma"
        .Interior.ColorIndex = 34
    End With

    ' Установить критерий замены
    With oCellReplaceFormat
        .Clear
        .Font.Name = "Arial"
        .Interior.ColorIndex = 35
    End With

    ' Выполнить замену
    wksAllowEditRange.UsedRange.Replace What:="", Replacement:="", _
                                        SearchFormat:=True, _
                                        ReplaceFormat:=True

    ' Сбросить формат поиска и замены
    oCellFindFormat.Clear
    oCellReplaceFormat.Clear

End Sub
```

Объект Characters

Объект Characters обеспечивает доступ к отдельным символам в текстовой строке. Символы обладают рядом визуальных свойств, модифицируемых с помощью этого объекта. В качестве родительских объектов для объекта Characters могут выступать AxisTitle, ChartTitle, DataLabel и Range. Каждый из родительских объектов предоставляет свойство Characters([Start], [Length]) для доступа к соответствующему фрагменту текста. Параметр Start указывает на начальный символ, а параметр Lenght описывает количество символов.

Общие свойства объекта Characters

Определения свойств Application, Creator и Parent приводились в начале приложения.

Свойства объекта Characters

Имя	*Тип возвращаемого значения*	*Описание*
Caption	String	Устанавливает/возвращает полную строку, которая содержится внутри объекта Characters
Count	Long	Только чтение. Возвращает количество символов, хранящихся в объекте
Font	Font	Только чтение. Возвращает объект, поддерживающий управление шрифтом символов
PhoneticCharacters	String	Устанавливает/возвращает фонетические символы, хранящиеся в объекте Characters
Text	String	Устанавливает/возвращает полную строку, содержащуюся в объекте Characters

Методы объекта Characters

Имя	*Тип возвращаемого значения*	*Параметры*	*Описание*
Delete	Variant		Удаляет символы из коллекции
Insert	Variant	String As String	Заменяет символы в коллекции на указанную строку

Пример: объект Characters

В этом примере для всех заглавных букв в активной ячейке устанавливается полужирный шрифт красного цвета размером 16 пунктов.

```
Sub FormatCellCapitals()
   Dim sText As String
   Dim oChars As Characters
   Dim i As Integer
   ' Получить текст из активной ячейки
   sText = ActiveCell.Text
   ' Перебрать символы текста
   For i = 1 To Len(sText)
      ' Является ли символ заглавным?
      If Asc(Mid(sText, i, 1)) > 64 _
         And Asc(Mid(sText, i, 1)) < 91 Then
```

```
        ' Да, поэтому получить объект Characters
        Set oChars = ActiveCell.Characters(i, 1)

        ' Установить красный цвет, размер 16 пунктов
        ' и полужирное выделение для объекта Characters.
        With oChars
            .Font.Color = RGB(255, 0, 0)
            .Font.Size = 16
            .Font.Bold = True
        End With
    End If
  Next
End Sub
```

Объект Chart и коллекция Charts

В коллекции Charts хранятся все листы диаграмм в пределах книги. Для этой коллекции в качестве родительского всегда выступает объект Workbook. В коллекции Charts хранятся только листы диаграмм. Отдельные диаграммы могут быть встроены в листы электронных таблиц и диалоговые листы. Для доступа к объекту Chart в пределах коллекции Charts можно воспользоваться свойством Item. В качестве параметра свойства Item указывается имя диаграммы или порядковый номер, описывающий положение диаграммы в книге (слева направо).

Объект Chart обеспечивает доступ ко всем атрибутам конкретной диаграммы в Excel. Это касается форматирования диаграммы, типа диаграммы и других ее свойств. Кроме этого, объект Chart предоставляет события, используемые для программного управления диаграммами.

Коллекция Charts предоставляет несколько свойств и методов, нехарактерных для остальных коллекций. Эти свойства и методы перечислены в следующей таблице.

Свойства и методы коллекции Charts

Имя	*Тип возвращаемого значения*	*Описание*
Count	Long	Только чтение. Возвращает количество диаграмм в коллекции
HPageBreaks	HPageBreaks	Только чтение. Возвращает коллекцию, содержащую все горизонтальные разрывы страницы, связанные с коллекцией Charts
VPageBreaks	VPageBreaks	Только чтение. Возвращает коллекцию, содержащую все вертикальные разрывы страниц, связанные с коллекцией Charts
Visible	Variant	Устанавливает/возвращает необходимость отображения диаграмм, входящих в коллекцию
Add	Chart	Метод. Параметры: [Before], [After], [Count]. Добавляет диаграмму в коллекцию. Точку назначения диаграммы можно определить, указав объект листа, который находится перед новым объектом диаграммы (параметр Before) или после нового объекта диаграммы (параметр After). Параметр Count определяет количество создаваемых диаграмм

Имя	*Тип возвращаемого значения*	*Описание*
Copy		Метод. Параметры: [Before], [After]. Добавляет копию активной диаграммы в точку, определенную с помощью параметров Before и After
Delete		Метод. Удаляет все диаграммы в коллекции
Move		Метод. Параметры: [Before], [After]. Перемещает текущую диаграмму в точку, определенную с помощью параметров
PrintOut		Метод. Параметры: [From], [To], [Copies], [Preview], [ActivePrinter], [PrintToFile], [Collate], [PrToFileName]. Печатает диаграммы из коллекции. Принтер, количество копий, порядок и необходимость предварительного просмотра определяются параметрами метода. Кроме этого, печать листов может быть перенаправлена в файл. Для этого можно воспользоваться параметрами PrintToFile и PrToFileName. Параметры From и To определяют диапазон печатаемых страниц
PrintPreview		Метод. Параметры: [EnableChanges]. Выводит текущую диаграмму из коллекции в режиме предварительного просмотра. Для отключения кнопок Поля (Margins) и Параметры (Setup) нужно установить параметр EnableChanges в значение False. При этом пользователь не сможет менять параметры страницы на этапе предварительного просмотра
Select		Метод. Параметры: [Replace]. Выделяет текущую диаграмму в коллекции

Общие свойства объекта Chart

Определения свойств Application, Creator и Parent приводились в начале приложения.

Свойства объекта Chart

Имя	*Тип возвращаемого значения*	*Описание*
Area3DGroup	ChartGroup	Только чтение. Возвращает объект ChartGroup, содержащий группу диаграммы с областями для трехмерной диаграммы
AutoScaling	Boolean	Устанавливает/возвращает необходимость изменения размера трехмерной диаграммы в соответствии с размерами двумерного эквивалента. При этом свойство RightAngleAxes должно быть установлено в значение True
Bar3DGroup	ChartGroup	Только чтение. Возвращает объект ChartGroup, содержащий группу гистограммы для трехмерной диаграммы

Имя	*Тип возвращаемого значения*	*Описание*
BarShape	XlBarShape	Устанавливает/возвращает базовую геометрическую форму, которая используется в трехмерной или двумерной гистограмме (например, параллелепипед, цилиндр, пирамида и т.д.)
ChartArea	ChartArea	Только чтение. Возвращает фрагмент диаграммы, содержащий оси, заголовки, легенды и свойства форматирования
ChartTitle	ChartTitle	Только чтение. Возвращает объект, поддерживающий управление свойствами заголовка диаграммы. Используется совместно со свойством HasTitle
ChartType	XlChartType	Устанавливает/возвращает тип диаграммы. Это свойство определяет действительность других свойств диаграммы. Например, если свойство ChartType установлено в значение xl3DbarClustered, то свойство Bar3DGroup может использоваться для получения доступа к свойствам группы диаграммы
CodeName	String	Только чтение. Возвращает программное имя диаграммы, определяемое на этапе проектирования в редакторе VBA
Column3D-Group	ChartGroup	Только чтение. Возвращает объект ChartGroup, содержащий группу гистограммы для трехмерной диаграммы
Corners	Corners	Только чтение. Возвращает объект, содержащий все углы трехмерной диаграммы
DataTable	DataTable	Только чтение. Возвращает объект, поддерживающий управление таблицей данных диаграммы
Depth-Percent	Long	Устанавливает/возвращает степень зависимости (в процентах) глубины трехмерной диаграммы (оси y) от ширины трехмерной диаграммы (оси x)
Display-BlanksAs	XlDisplay-BlanksAs	Устанавливает/возвращает способ обработки пустых ячеек при построении диаграммы (например, xlNotPlotted, xlZero или xlInterpolated)
Elevation	Long	Устанавливает/возвращает угол (в градусах), под которым пользователь видит трехмерную диаграмму. Допустимое значение угла зависит от типа трехмерной диаграммы
Floor	Floor	Только чтение. Возвращает объект с параметрами форматирования базового уровня трехмерной диаграммы
GapDepth	Long	Устанавливает/возвращает зависимость (в процентах) глубины рядов данных от ширины маркеров
HasAxis	Variant	Параметры: [Index1], [Index2]. Устанавливает/возвращает необходимость создания оси для диаграммы. Параметры могут использоваться для определения типа оси (с помощью констант xlAxisType в качестве значения первого параметра) и группы оси (с помощью констант xlAxisGroup в качестве значения второго параметра)
HasDataTable	Boolean	Устанавливает/возвращает наличие таблицы данных (если таблица данных связана, она отображается). Используется совместно со свойством DataTable

Имя	*Тип возвращаемого значения*	*Описание*
HasLegend	Boolean	Устанавливает/возвращает необходимость отображения легенды на диаграмме. Используется совместно со свойством Legend
HasPivot-Fields	Boolean	Устанавливает/возвращает необходимость отображения элементов управления сводной диаграммы. Свойство может устанавливаться в значение True только при использовании отчета сводной диаграммы
HasTitle	Boolean	Устанавливает/возвращает необходимость отображения заголовка диаграммы. Применяется совместно со свойством ChartTitle
Height-Percent	Long	Устанавливает/возвращает зависимость (в процентах) высоты трехмерной диаграммы (ось z) от ширины (ось x)
Hyperlinks	Hyperlinks	Только чтение. Возвращает коллекцию гиперссылок, связанных с диаграммой
Index	Long	Только чтение. Возвращает расположение диаграммы в родительской коллекции
Legend	Legend	Только чтение. Возвращает свойства форматирования для легенды. Используется совместно со свойством HasLegend
Line3DGroup	ChartGroup	Только чтение. Возвращает объект ChartGroup, содержащий группу диаграммы графика для трехмерной диаграммы
MailEnvelope	MsoEnvelope	Устанавливает/возвращает заголовок электронной почты для документа
Name	String	Устанавливает/возвращает имя диаграммы
Next	Object	Только чтение. Возвращает следующий лист в книге (слева направо) в виде объекта
PageSetup	PageSetup	Только чтение. Возвращает объект, поддерживающий управление параметрами страницы для диаграммы
Perspective	Long	Устанавливает угол перспективы, под которым отображается трехмерная диаграмма, если свойство RightAngle-Axes установлено в значение False
Pie3DGroup	ChartGroup	Только чтение. Возвращает объект ChartGroup, содержащий группу круговой диаграммы для трехмерной диаграммы
PivotLayout	PivotLayout	Только чтение. Возвращает объект, поддерживающий управление расположением полей отчета сводной диаграммы
PlotArea	PlotArea	Только чтение. Возвращает объект, поддерживающий управление форматированием, направляющими, маркерами данных и другими визуальными элементами области, в которой выполняется построение диаграммы. Эта область находится внутри области диаграммы
PlotBy	XlRowCol	Устанавливает/возвращает необходимость использования в качестве рядов данных столбцов оригинальных данных (значение xlColumns) или строк оригинальных данных (значение xlRows)

Имя	*Тип возвращаемого значения*	*Описание*
PlotVisible-Only	Boolean	Устанавливает/возвращает необходимость построения только видимых ячеек или видимых и скрытых ячеек (значение False)
Previous		Только чтение. Возвращает объект, представляющий предыдущий лист в книге (слева направо)
Protect-Contents	Boolean	Только чтение. Возвращает состояние защиты от изменений диаграммы и всего содержимого диаграммы
ProtectData	Boolean	Устанавливает/возвращает возможность перенаправления исходных данных для диаграммы
ProtectDrawingObjects	Boolean	Только чтение. Возвращает возможность модификации геометрических фигур в пределах диаграммы (ProtectDrawingObjects = False)
Protect-Formatting	Boolean	Устанавливает/возвращает возможность изменения параметров форматирования диаграммы
Protect-GoalSeek	Boolean	Устанавливает/возвращает возможность модификации точек на диаграмме с помощью указателя мыши
Protection-Mode	Boolean	Только чтение. Возвращает необходимость защиты пользовательского интерфейса. Даже если на пользовательский интерфейс диаграммы установлена защита, код VBA, связанный с диаграммой, может быть модифицирован
Protect-Selection	Boolean	Устанавливает/возвращает возможность выделения геометрических фигур в пределах диаграммы, а также возможность их добавления в диаграмму
RightAngle-Axes	Variant	Устанавливает/возвращает необходимость фиксации осей с использованием правильных углов для трехмерной диаграммы, даже если перспективная проекция диаграммы изменится
Rotation	Variant	Устанавливает/возвращает угол поворота относительно оси z (в градусах), под которым пользователь видит трехмерную диаграмму. Допустимые значения угла зависят от типа трехмерной диаграммы
Scripts	Scripts	Только чтение. Возвращает коллекцию кода VBScript, связанного с диаграммой (обычно этот код в дальнейшем используется на Web-страницах)
Shapes	Shapes	Только чтение. Возвращает все геометрические фигуры в пределах диаграммы
ShowWindow	Boolean	Устанавливает/возвращает возможность отображения встроенной диаграммы в отдельном окне, а не в виде встроенного объекта в пределах листа
SizeWith-Window	Boolean	Устанавливает/возвращает необходимость автоматического изменения размера листа диаграммы в соответствии с размером окна
SurfaceGroup	ChartGroup	Только чтение. Возвращает объект ChartGroup, содержащий группу диаграммы поверхности для трехмерной диаграммы
Tab	Tab	Только чтение. Возвращает объект Tab для диаграммы или листа

Имя	*Тип возвращаемого значения*	*Описание*
Visible	XlSheetVisibility	Устанавливает/возвращает необходимость отображения диаграммы. Свойство Visible может устанавливаться в значение xlVeryHidden. В таком случае диаграмма окажется недоступной для конечного пользователя
Walls	Walls	Только чтение. Возвращает объект, позволяющий управлять гранями трехмерной диаграммы
WallsAnd-Gridlines2D	Boolean	Устанавливает/возвращает необходимость двумерного отображения линий и граней трехмерной гистограммы, трехмерной диаграммы с областями и трехмерной кластерной диаграммы

Методы объекта Chart

Имя	*Тип возвращаемого значения*	*Параметры*	*Описание*
Activate			Активизирует диаграмму. После этого на диаграмму указывает свойство ActiveChart
Apply-CustomType		ChartType As XlChartType, [TypeName]	Меняет тип диаграммы на тип, указанный в качестве значения параметра ChartType. Если параметр ChartType равен xlUserDefined, второй параметр позволяет указать собственное название типа диаграммы
Apply-DataLabels		[Type As XlDataLabelsType], [LegendKey], [AutoText], [HasLeaderLines], [ShowSeriesName], [ShowCategoryName], [ShowValue], [ShowPercentage], [ShowBubbleSize], [Separator]	Устанавливает маркеры точек для диаграммы. В параметре Type указывается необходимость отображения метки категории, процента от целого, значения или отказа от вывода метки. Возле метки может отображаться ключ легенды. Для этого параметр LegendKey необходимо установить в значение True
AreaGroups	Object	[Index]	Возвращает единственную группу диаграммы с областями (ChartGroup) или коллекцию групп диаграммы с областями (ChartGroups) для двумерной диаграммы
Axes	Object	Type, AxisGroup As XlAxisGroup	Возвращает объект Axis или коллекцию Axes для соответствующей диаграммы. Тип оси и группы осей могут выбираться с помощью параметров

Имя	*Тип возвращаемого значения*	*Параметры*	*Описание*
BarGroups	Object	[Index]	Возвращает единственную группу линейчатой диаграммы (ChartGroup) или коллекцию групп линейчатой диаграммы (ChartGroups) для двумерной диаграммы
ChartGroups	Object	[Index]	Возвращает единственную группу диаграммы (ChartGroup) или коллекцию групп диаграммы (ChartGroups) для диаграммы
ChartObjects	Object	[Index]	Возвращает единственную встроенную диаграмму (ChartObject) или коллекцию встроенных диаграмм (ChartObjects) в пределах диаграммы
ChartWizard		[Source], [Gallery], [Format], [PlotBy], [CategoryLabels], [SeriesLabels], [HasLegend], [Title], [CategoryTitle], [ValueTitle], [ExtraTitle]	Единый метод для модификации ключевых свойств диаграммы. Укажите свойства, значения которых необходимо изменить. Параметр Source позволяет указать источник данных. Параметр Gallery позволяет указать тип диаграммы. Параметр Format позволяет указать один из десяти встроенных автоформатов диаграммы. Остальные параметры указывают способ получения данных из источника, источник меток категорий, источник меток рядов, необходимость отображения легенды, а также заголовки диаграммы и осей. Если значение параметра Source не указано, этот метод может использоваться, только если содержащий диаграмму лист является активным
CheckSpelling		[CustomDictionary], [IgnoreUppercase], [AlwaysSuggest], [SpellLang]	Проверяет орфографию текста на диаграмме. При этом можно указать собственный словарь (CustomDictionary). Кроме этого, можно игнорировать все слова в верхнем регистре (IgnoreUppercase). Excel можно заставить отображать список предложенных замен (AlwaysSuggest)

Имя	*Тип возвращаемого значения*	*Параметры*	*Описание*
ColumnGroups	Object	[Index]	Возвращает единственную группу гистограммы (ChartGroup) или коллекцию групп гистограммы (ChartGroups) для двумерной диаграммы
Copy		[Before], [After]	Добавляет новую копию диаграммы. Положение определяется параметрами Before или After
CopyPicture		[AppearanceAs XlPicture-Appearance], [Format As XlCopyPicture-Format], [Size As XlPicture-Appearance]	Копирует диаграмму в буфер обмена в виде изображения. Параметр Appearance позволяет определить копирование экранного представления или представления для печати. Параметр Format позволяет определить тип изображения, которое копируется в буфер обмена. Параметр Size дает возможность определить размер изображения при копировании диаграммы с листа диаграммы
CreatePub-lisher		Edition, Appearance As XlPic-tureAppearance, Size As XlPic-tureAppearance, [ContainsPICT], [ContainsBIFF], [ContainsRTF], [ContainsVALU]	Используется на компьютерах Macintosh для создания изображения диаграммы в стандартном формате. Метод является эквивалентом метода CopyPicture на ПК
Delete			Удаляет диаграмму
Deselect			Снимает выделение с объекта диаграммы
Doughnut-Groups	Object	[Index]	Возвращает единственную группу кольцевой диаграммы (ChartGroup) или коллекцию групп кольцевой диаграммы (ChartGroups) для двумерной диаграммы
Evaluate	Variant	Name	Оценивает значение параметра Name так, как будто строка вводилась в качестве выражения в ячейку листа
Export	Boolean	Filename As String, [FilterName], [Interactive]	Сохраняет диаграмму в виде изображения (в формате jpg или gif) в файле, имя которого передается в качестве значения параметра Filename

Имя	*Тип возвращаемого значения*	*Параметры*	*Описание*
GetChart-Element		x As Long, y As Long, ElementID As Long, Arg1 As Long, Arg2 As Long	Возвращает элемент диаграммы, расположенный по координатам x и y. Передаются только первые два параметра. Переменные должны передаваться в качестве значений последних трех параметров. После вызова метода в последних трех параметрах можно ожидать возвращаемых значений. Параметр ElementID возвращает одно из значений xlChartItem. Параметры Arg1 и Arg2 могут содержать или не содержать данные в зависимости от типа элемента
LineGroups	Object	[Index]	Возвращает единственную группу диаграммы графика (ChartGroup) или коллекцию групп диаграммы графика (ChartGroups) для двумерной диаграммы
Location	Chart	Where As XlChart-Location, [Name]	Перемещает диаграмму в позицию, указанную в качестве значений параметров Where и Name. Параметр Where позволяет указать точку назначения перемещаемой диаграммы: лист диаграммы или встроенную диаграмму
Move		[Before], [After]	Перемещает диаграмму в точку, определенную в параметрах
OLEObjects	Object	[Index]	Возвращает единственный объект OLE (OLEObject) или коллекцию объектов OLE (OLEObjects) для диаграммы
Paste&		[Type]	Вставляет данные или изображение из буфера обмена в диаграмму. Параметр Type может использоваться для выбора вставки форматов, формул или всего содержимого буфера обмена
PieGroups	Object	[Index]	Возвращает единственную группу круговой диаграммы (ChartGroup) или коллекцию групп круговой диаграммы (ChartGroups) для двумерной диаграммы

Имя	*Тип возвращаемого значения*	*Параметры*	*Описание*
PrintOut		[From], [To], [Copies], [Preview], [ActivePrinter], [PrintToFile], [Collate], [PrToFileName]	Выводит диаграмму на печать. Принтер, количество копий, порядок сортировки и необходимость предварительного просмотра указываются с помощью параметров метода. Кроме этого, лист можно распечатать в файл. Для этого используются параметры PrintToFile и PrToFileName. Параметры From и To могут использоваться для определения диапазона печатаемых страниц
PrintPreview		[EnableChanges]	Выводит текущую диаграмму в режиме предварительного просмотра. Для отключения кнопок Поля (Margins) и Параметры страницы (Setup) параметр EnableChanges необходимо установить в значение False. При этом пользователь не сможет изменить параметры страницы
Protect		[Password], [DrawingObjects], [Contents], [Scenarios], [UserInterfaceOnly]	Защищает диаграмму от изменений. При этом можно указать чувствительный к регистру пароль в качестве значения свойства Password. Кроме этого, метод позволяет защитить геометрические фигуры (DrawingObjects), все содержимое диаграммы (Contents) или только пользовательский интерфейс (UserInterfaceOnly)
RadarGroups	Object	[Index]	Возвращает единственную группу лепестковой диаграммы (ChartGroup) или коллекцию групп лепестковой диаграммы (ChartGroups) для двумерной диаграммы
Refresh			Обновляет содержимое диаграммы, перечитывая источник данных

Имя	*Тип возвращаемого значения*	*Параметры*	*Описание*
SaveAs		Filename As String, [FileFormat], [Password], [WriteResPassword], [ReadOnly-Recommended], [CreateBackup], [AddToMru], [TextCodepage], [TextVisualLayout], [Local]	Сохраняет текущую диаграмму в новой книге. Имя файла указывается в качестве значения параметра Filename. Формат файла, пароль, пароль для чтения, создание резервных копий и другие свойства сохраняемого файла указываются с помощью параметров метода
Select		[Replace]	Выделяет диаграмму
Series-Collection	Object	[Index]	Возвращает единственный ряд (Series) или коллекцию рядов (SeriesCollection) для диаграммы
Set-Background-Picture		FileName As String	Устанавливает фон диаграммы с использованием изображения из файла, имя которого передается в качестве значения параметра FileName
SetSource-Data		Source As Range, [PlotBy]	Устанавливает в качестве источника данных диаграммы диапазон, на который указывает параметр Source. В качестве значений параметра PlotBy используются константы XlRowCol. Этот параметр позволяет выбрать построение строк или столбцов
Unprotect		[Password]	Снимает защиту с диаграммы. Если диаграмма защищена паролем, пароль указывается в качестве параметра метода
XYGroups	Object	[Index]	Возвращает единственную группу точечной диаграммы (ChartGroup) или коллекцию групп точечной диаграммы (ChartGroups) для двумерной диаграммы

События объекта Chart

Имя	*Тип возвращаемого значения*	*Описание*
Activate		Возникает при передаче фокуса в диаграмму
Before-DoubleClick	ElementID As XlChartItem, Arg1 As Long, Arg2 As Long, Cancel As Boolean	Возникает перед двойным щелчком на диаграмме. Элемент, на котором выполняется двойной щелчок, передается в процедуру обработки события в качестве значения параметра ElementID. Параметры Arg1 и Arg2 могут содержать или не содержать значения в зависимости от значения параметра ElementID. Событие двойного щелчка можно отменить. Для этого параметр Cancel устанавливается в значение True

Имя	*Тип возвращаемого значения*	*Описание*
`BeforeRightClick`	`Cancel As Boolean`	Возникает перед щелчком правой кнопкой мыши на диаграмме. Событие щелчка правой кнопкой мыши можно отменить. Для этого параметр `Cancel` необходимо установить в значение `True`
`Calculate`		Возникает после отображения новых или измененных данных в диаграмме
`Deactivate`		Возникает при потере фокуса диаграммой
`DragOver`		Возникает, когда в область диаграммы перетаскивается диапазон ячеек. Обычно событие используется для изменения курсора мыши или вывода сообщения о текущем состоянии
`DragPlot`		Возникает при перетаскивании и отпускании диапазона ячеек в области диаграммы. Обычно событие используется для модификации атрибутов диаграммы
`MouseDown`	`Button As XlMouseButton, Shift As Long, x As Long, y As Long`	Возникает при нажатии кнопки мыши в области диаграммы. Каждая нажатая кнопка мыши передается в значении параметра `Button`. Параметр `Shift` содержит информацию о состоянии клавиш <Ctrl>, <Shift> и <Alt>. Через параметры `x` и `y` передаются координаты указателя мыши
`MouseMove`	`Button As XlMouseButton, Shift As Long, x As Long, y As Long`	Возникает при наведении указателя мыши на диаграмму. Каждая нажатая кнопка мыши передается в значении параметра `Button`. Параметр `Shift` содержит информацию о состоянии клавиш <Ctrl>, <Shift> и <Alt>. Через параметры `x` и `y` передаются координаты указателя мыши
`MouseUp`	`Button As XlMouseButton, Shift As Long, x As Long, y As Long`	Возникает при отпускании кнопки мыши над диаграммой. Каждая нажатая кнопка мыши передается в значении параметра `Button`. Параметр `Shift` содержит информацию о состоянии клавиш <Ctrl>, <Shift> и <Alt>. Через параметры `x` и `y` передаются координаты указателя мыши
`Resize`		Возникает при изменении размеров диаграммы
`Select`	`ElementID As XlChartItem, Arg1 As Long, Arg2 As Long`	Возникает при выделении элемента диаграммы. Выделенный элемент передается в процедуру обработки события в качестве значения параметра `ElementID`. Параметры `Arg1` и `Arg2` могут содержать или не содержать значения в зависимости от значения параметра `ElementID`
`SeriesChange`	`SeriesIndex As Long, PointIndex As Long`	Возникает при изменении значения точки на диаграмме. Параметр `SeriesIndex` содержит расположение ряда в коллекции рядов диаграммы. Параметр `PointIndex` возвращает расположение точки в пределах ряда

Пример: объект Chart и коллекция Charts

В этом примере на основе таблицы, содержащей активную ячейку, создается трехмерная диаграмма, выполняется ее форматирование, и изображение диаграммы сохраняется в формате `.jpg`:

```
Sub CreateAndExportChart()
  Dim oCht As Chart
  ' Создать новую (пустую) диаграмму
  Set oCht = Charts.Add
```

```
  ' Отформатировать диаграмму
  With oCht
     .ChartType = xl3DColumnStacked
     '  Установить источник данных  и построить данные из столбцов
     .SetSourceData Source:=Selection.CurrentRegion, _
                    PlotBy:=xlColumns

     ' Создать новый лист для диаграммы
     .Location Where:=xlLocationAsNewSheet
     ' Размер и форма должны соответствовать размерам и форме окна
     .SizeWithWindow = True
     ' Отключить растягивание диаграммы
     .AutoScaling = False
     ' Настроить диаграмму
     .HasTitle = True
     .ChartTitle.Caption = "Основная диаграмма"
     ' Отсутствие заголовков для осей
     .Axes(xlCategory).HasTitle = False
     .Axes(xlSeries).HasTitle = False
     .Axes(xlValue).HasTitle = False
     ' Включить трехмерное представление диаграммы
     .RightAngleAxes = False
     .Elevation = 50   'degrees
     .Perspective = 30 'degrees
     .Rotation = 20    'degrees
     .HeightPercent = 100
     ' Метки данных не должны отображаться
     .ApplyDataLabels Type:=xlDataLabelsShowNone
     '  Сохранить изображение диаграммы в формате jpg
     .Export "c:\" &.Name &".jpg", "jpg", False
  End With
End Sub
```

Объект ChartArea

Этот объект предоставляет параметры форматирования, связанные с областью диаграммы. Для двумерных диаграмм объект содержит оси, заголовки осей и диаграмм. Для трехмерных диаграмм объект содержит заголовок и легенду диаграммы. Часть диаграммы, в которой отображаются данные (область построения), не входит в объект ChartArea. Дополнительная информация о форматировании области построения приводится в разделе, посвященном объекту PlotArea. Родительским для объекта ChartArea всегда является объект Chart.

Общие свойства объекта ChartArea

Определения свойств Application, Creator и Parent приводились в начале приложения.

Свойства объекта ChartArea

Имя	*Тип возвращаемого значения*	*Описание*
AutoScaleFont	Variant	Устанавливает/возвращает необходимость изменения размера шрифта в объекте ChartArea при изменении размера в объекте Chart
Border	Border	Только чтение. Возвращает атрибуты границ вокруг выделенной области диаграммы

Имя	*Тип возвращаемого значения*	*Описание*
Fill	ChartFill-Format	Только чтение. Возвращает объект, поддерживающий управление атрибутами заполнения области диаграммы
Font	Font	Только чтение. Предоставляет доступ к таким свойствам Font, как Type и Size
Height	Double	Устанавливает/возвращает высоту области диаграммы в пунктах
Interior	Interior	Только чтение. Возвращает объект, поддерживающий управление форматированием внутренней части области диаграммы (например, цветом внутренней области)
Left	Double	Устанавливает/возвращает расположение левого края области диаграммы относительно диаграммы (в пунктах)
Name	String	Только чтение. Возвращает имя области диаграммы
Shadow	Boolean	Устанавливает/возвращает необходимость создания эффекта тени вокруг области диаграммы
Top	Double	Устанавливает/возвращает расположение верхнего края области диаграммы относительно диаграммы (в пунктах)
Width	Double	Устанавливает/возвращает ширину области диаграммы в пунктах

Методы объекта ChartArea

Имя	*Тип возвращаемого значения*	*Параметры*	*Описание*
Clear	Variant		Очищает область диаграммы
ClearContents	Variant		Удаляет содержимое из области диаграммы, не затрагивая форматирование
ClearFormats	Variant		Удаляет форматирование из области диаграммы, не затрагивая данные
Copy	Variant		Копирует область диаграммы в буфер обмена
Select	Variant		Активизирует и выделяет область диаграммы

Пример: объект ChartArea

Настройка форматирования области диаграммы:

```
Sub FormatChartArea()
   Dim oCA As ChartArea
   Set oCA = Charts(1).ChartArea
   With oCA
      .Border.LineStyle = xlContinuous
      .Fill.PresetTextured msoTextureCanvas
      .Fill.Visible = msoTrue
   End With
End Sub
```

Объект ChartColorFormat

Этот объект описывает цвет для родительского объекта ChartFillFormat. Например, объект ChartFillFormat предоставляет свойство BackColor, которое возвращает объект ChartColorFormat, позволяющий устанавливать цвет.

Общие свойства объекта ChartColorFormat

Определения свойств Application, Creator и Parent приводились в начале приложения.

Свойства объекта ChartColorFormat

Имя	*Тип возвращаемого значения*	*Описание*
RGB	Long	Только чтение. Возвращает значения красного, зеленого и синего цветов, позволяющие сформировать интересующий цвет
SchemeColor	Long	Устанавливает/возвращает цвет для объекта ChartColorFormat с помощью номера индекса в соответствующей цветовой схеме
Type	Long	Только чтение. Возвращает тип цвета: RGB, смешанный или цвет из схемы

Пример: объект ChartColorFormat

В этом примере в качестве шаблона заливки диаграммы используется встроенный цвет номер 6. После этого выводятся значения красного, зеленого и синего цветов для выбранного встроенного цвета.

```
Sub SetChartColorFormat()
   Dim oCCF As ChartColorFormat
   With Charts(3).PlotArea.Fill
      ' Удостоверится, что используется шаблон Fill
      .Visible = True

      ' Получить объект ChartColorFormat для объекта ForeColor
      Set oCCF = .ForeColor
      ' Выбрать встроенный цвет #6
      oCCF.SchemeColor = 6

      ' Считать значения красного, зеленого и синего цветов
      ' для цвета #6
      MsgBox "ForeColor #6 RGB is:" & vbCrLf & _
      "Красный = " & ((oCCF.RGB And &HFF0000) / &H10000) & vbCrLf & _
      "Зеленый = " & ((oCCF.RGB And &HFF00) / &H100) & vbCrLf & _
      "Синий = " & ((oCCF.RGB And &HFF))
   End With
End Sub
```

Объект ChartFillFormat

Этот объект описывает форматирование заполнения, связанное с родительским объектом. Он также поддерживает управление основным цветом, цветом фона, а также шаблонами, связанными с родительским объектом.

Общие свойства ChartFillFormat

Определения свойств Application, Creator и Parent приводились в начале приложения.

Свойства объекта ChartFillFormat

Имя	*Тип возвращаемого значения*	*Описание*
BackColor	ChartColorFormat	Только чтение. Возвращает цвет фона в виде объекта ChartColorFormat
ForeColor	ChartColorFormat	Только чтение. Возвращает основной цвет в виде объекта ChartColorFormat
GradientColorType	MsoGradientColorType	Только чтение. Возвращает тип используемого градиентного заполнения цветом
GradientDegree	Single	Только чтение. Возвращает степень яркости градиентной заливки
GradientStyle	MsoGradientStyle	Только чтение. Возвращает используемую ориентацию градиентной заливки
GradientVariant	Long	Только чтение. Возвращает применяемый центральный градиент
Pattern	MsoPatternType	Только чтение. Возвращает шаблон, который используется для заполнения
PresetGradientType	MsoPresetGradientType	Только чтение. Возвращает тип применяемого градиента
PresetTexture	MsoPresetTexture	Только чтение. Возвращает встроенную текстуру заполнения
TextureName	String	Только чтение. Возвращает невстроенную текстуру заполнения
TextureType	MsoTextureType	Только чтение. Возвращает информацию о том, является ли текстура собственной, встроенной или смешанной
Type	MsoFillType	Устанавливает/возвращает прозрачность заполнения. Допустимые значения находятся в пределах от 0 (непрозрачная) до 1 (полностью прозрачная)
Visible	MsoTriState	Только чтение. Возвращает информацию о том, является ли заполнение текстурой, градиентом, сплошным цветом, фоном, изображением или смешанным заполнением

Методы объекта ChartFillFormat

Имя	*Тип возвращаемого значения*	*Параметры*	*Описание*
OneColorGradient		Style As MsoGradientStyle, Variant As Long, Degree As Single	Устанавливает стиль, вариант и степень градиентного заполнения одним цветом
Patterned		Pattern As MsoPatternType	Устанавливает шаблон заполнения

Имя	*Тип возвращаемого значения*	*Параметры*	*Описание*
`PresetGradient`		`Style As MsoGradientStyle, Variant As Long, PresetGradientType As MsoPresetGradientType`	Выбирает стиль, вариант и встроенный тип градиента для градиентного заполнения
`PresetTextured`		`PresetTexture As MsoPresetTexture`	Устанавливает встроенную текстуру для заполнения
`Solid`			Устанавливает заполнение одним цветом
`TwoColorGradient`		`Style As MsoGradientStyle, Variant As Long`	Устанавливает заполнение двухцветным градиентом
`UserPicture`		`[PictureFile], [PictureFormat], [PictureStackUnit], [Picture Placement]`	Устанавливает заполнение с помощью изображения в формате `PictureFile`
`UserTextured`		`TextureFile As String`	Устанавливает заполнение с помощью собственной текстуры в формате `TextureFormat`

Пример: объект ChartFillFormat

```
Sub FormatPlotArea()
   Dim oCFF As ChartFillFormat
   ' Получить объект ChartFillFormat для области построения
   Set oCFF = ActiveSheet.ChartObjects(1).Chart.PlotArea.Fill
   ' Установить формат области заполнения
   With oCFF
       .TwoColorGradient Style:=msoGradientDiagonalUp, Variant:=1
       .Visible = True
       .ForeColor.SchemeColor = 6
       .BackColor.SchemeColor = 7
   End With
End Sub
```

Объект ChartGroup и коллекция ChartGroups

В коллекции `ChartGroups` хранится информация о построении диаграммы, связанная с родительской диаграммой. Диаграмма может содержать больше одного объекта `ChartGroup`. Например, одна диаграмма может быть связана одновременно с диаграммой графика и линейчатой диаграммой. Свойство `ChartGroups` объекта `Chart` может использоваться для получения доступа к коллекции `ChartGroups`. Кроме этого, свойства `PieGroups` и `LineGroups` объекта `Chart` также возвращают только группы круговой диаграммы и диаграммы графика соответственно.

Кроме обычных свойств коллекции, коллекция `ChartGroups` предоставляет свойство `Count`, которое возвращает количество объектов `ChartGroup` в пределах коллекции. Родительским для коллекции `ChartGroups` или объекта `ChartGroup` является объект `Chart`.

Объект `ChartGroup` содержит все отображаемые точки диаграммы соответствующего типа. Он также может содержать несколько рядов (каждый столбец или строка исходных данных). Каждый ряд может содержать несколько точек (для каждой ячейки исходных данных). С объектом `Chart` может быть связано больше одного объекта `Chart-`

Group. Свойства Bar3DGroup, Column3DGroup, Line3DGroup, Pie3Dgroup и SurfaceGroup объекта Chart позволяют получить доступ к определенной группе, соответствующей типу диаграммы. Методы AreaGroups, BarGroups, ColumnGroups, DoughnutGroups, LineGroups, PieGroups, RadarGroups и XYGroups объекта Chart позволяют получать объекты ChartGroup или коллекции ChartGroups.

Общие свойства ChartGroup

Определения свойств Application, Creator и Parent приводились в начале приложения.

Свойства объекта ChartGroup

Имя	*Тип возвращаемого значения*	*Описание*
AxisGroup	XlAxisGroup	Устанавливает/возвращает, является ли группа диаграммы основной или дополнительной
BubbleScale	Long	Устанавливает/возвращает процент увеличения размера точек относительно принятого по умолчанию размера. Действительные значения находятся в пределах 0 от 300. Это свойство может использоваться только для группы пузырьковой диаграммы
DoughnutHoleSize	Long	Устанавливает/возвращает размер отверстия в группе круговой диаграммы. В качестве значения указывается относительный размер в процентах. Действительные значения входят в диапазон от 10 до 90. Это свойство может использоваться только вместе с круговыми диаграммами
DownBars	DownBars	Только чтение. Возвращает объект, позволяющий управлять форматированием полос понижения на группе диаграммы графика. Это свойство действительно только для групп диаграммы графика. Используется совместно со свойством HasUpDownBars
DropLines	DropLines	Только чтение. Возвращает объект, позволяющий управлять параметрами форматирования линий в группе диаграммы графика или диаграммы с областями. Свойство действительно только для групп диаграммы графика или групп диаграммы с областями. Используется совместно со свойством HasDropLines
FirstSliceAngle	Long	Устанавливает/возвращает угол, который используется для первого сектора групп круговой или кольцевой диаграммы (первая точка данных, отображаемая на экране)
GapWidth	Long	Устанавливает/возвращает размер промежутка между столбцами различных последовательностей данных. Кроме этого, при работе с линейчатыми/круговыми или круговыми/круговыми диаграммами свойство GapWidth описывает расстояние между основной и дополнительной диаграммами (свойство ChartType родительской диаграммы должно быть равно xlPieOfPie или xlBarOfPie)
Has3Dshading	Boolean	Устанавливает/возвращает необходимость применения трехмерных эффектов при отображении элементов управления группы диаграммы

Имя	*Тип возвращаемого значения*	*Описание*
HasDropLines	Boolean	Устанавливает/возвращает необходимость отображения линий при выводе группы диаграммы. Это свойство используется совместно со свойством DownLines
HasHiLoLines	Boolean	Устанавливает/возвращает необходимость отображения линий максимального и минимального уровней при построении группы диаграммы. Это свойство применяется совместно со свойством HiLoLines
HasRadar-AxisLabels	Boolean	Устанавливает/возвращает необходимость отображения меток углов на угловой диаграмме. Свойство действительно только для групп угловой диаграммы
HasSeries-Lines	Boolean	Устанавливает/возвращает необходимость отображения линий рядов для группы диаграммы. Это свойство используется совместно со свойством SeriesLines
HasUpDown-Bars	Boolean	Устанавливает/возвращает необходимость отображения полос повышения и понижения для группы диаграммы. Это свойство используется совместно со свойством DownBars и UpBars
HiLoLines	HiLoLines	Только чтение. Возвращает объект, позволяющий управлять форматированием линий максимального-минимального уровня в диаграмме графика. Это свойство действительно только для диаграмм графика
Index	Long	Только чтение. Возвращает положение текущего объекта ChartGroup в родительской коллекции
Overlap	Long	Устанавливает/возвращает необходимость перекрытия или разделения полос и столбцов внутри рядов. Свойству могут присваиваться значения в диапазоне от -100 до 100, где -100 приведет к отображению промежутка между полосами/столбцами, равного ширине полосы/столбца. Значение 1000 приводит к наложению полос/диаграмм друг на друга. Это свойство действительно только для групп двумерных линейчатых диаграмм и гистограмм
RadarAxis-Labels	TickLabels	Только чтение. Возвращает объект, позволяющий управлять форматированием и метками осей угловой диаграммы. Это свойство действительно только для групп угловой диаграммы
SecondPlot-Size	Long	Устанавливает/возвращает соотношение (в процентах) между размером дополнительной группы кругово/круговой или линейчатой/круговой диаграммы и размером основной диаграммы
SeriesLines	SeriesLines	Только чтение. Возвращает объект, позволяющий управлять форматированием линий рядов в группе диаграммы. Линия ряда соединяет одинаковые ряды данных в группах наложенных гистограмм, наложенных линейчатых диаграмм, круговой/круговой или линейчатой/круговой. Это свойство используется совместно со свойством HasSeriesLines
ShowNega-tiveBubbles	Boolean	Устанавливает/возвращает необходимость отображения точек с отрицательными значениями. Это свойство действительно только для групп пузырьковой диаграммы

Имя	*Тип возвращаемого значения*	*Описание*
`SizeRepresents`	`XlSizeRepresents`	Устанавливает/возвращает необходимость представления значения точки данных с помощью размера или области точки на группе пузырьковой диаграммы. Это свойство действительно только для группы пузырьковой диаграммы
`SplitType`	`XlChartSplitType`	Устанавливает/возвращает способ разделения двух диаграмм в группе круговой/круговой или линейчатой/круговой. Например, диаграмма может делиться по процентному значению (`xlSplitByPercentValue`) или по значению (`xlSplitByValue`)
`SplitValue`	`Variant`	Устанавливает/возвращает значение, которое комбинируется с основной круговой диаграммой и рассматривается подробно в дополнительной диаграмме в группе круговой/круговой или линейчатой/круговой диаграммы
`UpBars`	`UpBars`	Возвращает объект, поддерживающий управление форматированием полос повышения в группе диаграммы графика. Это свойство действительно только для групп диаграммы графика. Свойство используется совместно со свойством `HasUpDownBars`
`VaryByCategories`	`Boolean`	Устанавливает/возвращает необходимость использования разных цветов при отображении различных категорий в одной последовательности группы диаграммы. Для использования разных цветов диаграмма должна содержать единственную последовательность данных

Методы объекта ChartGroup

Имя	*Тип возвращаемого значения*	*Параметры*	*Описание*
`SeriesCollection`	`Object`	`[Index]`	Возвращает единственный ряд (`Series`) или коллекцию рядов (`SeriesCollection`) для диаграммы

Пример: объект ChartGroup и коллекция ChartGroups

Этот код устанавливает промежуток для всех групп гистограммы в пределах диаграммы равным 10% и включает окрашивание каждого гистобара в свой цвет.

```
Sub FormatColumns()
   Dim oCht As Chart
   Dim oCG As ChartGroup
   For Each oCG In Charts(1).ColumnGroups
      oCG.GapWidth = 10
      oCG.VaryByCategories = True
   Next
End Sub
```

Объект ChartObject и коллекция ChartObjects

Коллекция `ChartObjects` содержит все встроенные в лист электронной таблицы, лист диаграммы или диалоговый лист объекты `Chart`. Сами листы диаграмм в этой коллекции не хранятся. Они доступны в коллекции `Charts`. Каждый объект `Chart` из коллекции `ChartObjects` доступен через объект `ChartObject`. При этом объект `Chart`-

Object выступает в качестве оболочки для диаграммы. Свойство Chart объекта Chart-Object позволяет получить доступ к диаграмме. Кроме этого, объект ChartObject предоставляет свойства для модификации форматирования встроенной диаграммы (например, Height и Width).

Коллекция ChartObjects предоставляет ряд свойств, выходящих за пределы типичных атрибутов коллекции. Эти свойства перечислены ниже.

Свойства и методы коллекции ChartObjects

Имя	*Тип возвращаемого значения*	*Описание*
Border	Border	Только чтение. Возвращает свойства границ вокруг коллекции объектов диаграммы
Count	Long	Только чтение. Возвращает количество объектов Chart-Object в коллекции
Enabled	Boolean	Устанавливает/возвращает возможность запуска пользователем макросов, связанных с каждым объектом Chart-Object в коллекции
Height	Double	Устанавливает/возвращает высоту объекта ChartObject, если в коллекции находится только один объект
Interior	Interior	Только чтение. Возвращает объект, содержащий параметры форматирования внутренней области всех объектов Chart в коллекции (например, цвета внутренней области)
Left	Double	Устанавливает/возвращает расстояние от левого края объекта ChartObject до левого края родительского листа. Это свойство рабоатет, когда в коллекции хранится только один объект ChartObject
Locked	Boolean	Устанавливает/возвращает необходимость блокирования объекта ChartObject при защите родительского листа. Это свойство работает, когда в коллекции хранится только один объект ChartObject
Placement	Variant	Устанавливает/возвращает способ привязки объекта ChartObject на листе (например, свободное расположение, перемещение вместе с ячейками и т.д.) Для установки этого свойства можно воспользоваться константами XlPlacement. Это свойство работает, если в коллекции хранится только один объект ChartObject
PrintObject	Boolean	Устанавливает/возвращает необходимость печати встроенной диаграммы при печати родительского листа. Это свойство работает, если в коллекции хранится только один объект ChartObject
Rounded-Corners	Boolean	Устанавливает/возвращает необходимость закругления углов встроенной диаграммы (True) или отображения правильных углов диаграммы (False). Это свойство работает, если в коллекции хранится только один объект ChartObject

Имя	*Тип возвращаемого значения*	*Описание*
Shadow	Boolean	Устанавливает/возвращает необходимость отображения эффекта тени вокруг встроенной диаграммы. Это свойство работает, если в коллекции хранится только один объект ChartObject
ShapeRange	ShapeRange	Только чтение. Возвращает объекты ChartObject из коллекции как объекты Shape
Top	Double	Устанавливает/возвращает расстояние от верхнего края объекта ChartObject до вершины родительского листа. Это свойство работает, если в коллекции хранится только один объект ChartObject
Visible	Boolean	Устанавливает/возвращает видимость всех объектов из коллекции ChartObjects
Width	Double	Устанавливает/возвращает ширину объекта ChartObject, если в коллекции хранится только один объект
Add	ChartObject	Метод. Параметры: Left As Double, Top As Double, Width As Double, Height As Double. Добавляет объект ChartObject в коллекцию ChartObjects. Положение нового объекта ChartObject указывается с помощью параметров Left, Top, Width и Height
BringToFront	Variant	Метод. Размещает все объекты коллекции ChartObjects на переднем плане поверх всех остальных объектов
Copy	Variant	Метод. Копирует все объекты ChartObject из коллекции в буфер обмена
CopyPicture	Variant	Метод. Параметры: Appearance As XlPictureAppearance, Format As XlCopyPictureFormat. Копирует изображения объектов Chart из коллекции в буфер обмена. Параметр Appearance позволяет выбрать копирование экранного представления или представления для печати. Параметр Format позволяет выбрать тип изображения, которое будет скопировано в буфер обмена
Cut	Variant	Метод. Вырезает все объекты ChartObject из коллекции в буфер обмена
Delete	Variant	Метод. Удаляет все объекты ChartObject из коллекции
Duplicate		Метод. Дублирует все объекты ChartObject из коллекции в родительском листе. (Например, если в родительском листе присутствует два объекта ChartObject, в результате вызова этого метода на листе окажется четыре объекта ChartObject.)
Select	Variant	Метод. Параметры: [Replace]. Выделяет все объекты ChartObject в коллекции
SendToBack	Variant	Метод. Размещает объекты ChartObject из коллекции на заднем плане за другими объектами

Общие свойства объекта ChartObject

Определения свойств Application, Creator и Parent приводились в начале приложения.

Свойства объекта ChartObject

Имя	*Тип возвращаемого значения*	*Описание*
Border	Border	Только чтение. Возвращает свойства границы вокруг встроенной диаграммы
BottomRightCell	Range	Только чтение. Возвращает диапазон, состоящий из одной ячейки и расположенный под нижним правым углом объекта ChartObject
Chart	Chart	Только чтение. Возвращает фактическую диаграмму, связанную с объектом ChartObject
Enabled	Boolean	Устанавливает/возвращает необходимость запуска макроса, связанного с объектом ChartObject
Height	Double	Устанавливает/возвращает высоту встроенной диаграммы
Index	Long	Только чтение. Возвращает положение объекта ChartObject в родительской коллекции
Interior	Interior	Только чтение. Возвращает объект, поддерживающий управление форматированием внутренней области объекта диаграммы (например, поддерживающий установку цвета внутренней области)
Left	Double	Устанавливает/возвращает расстояние от левого края объекта ChartObject до левого края родительского листа
Locked	Boolean	Устанавливает/возвращает необходимость блокирования объекта ChartObject при защите родительского листа
Name	String	Устанавливает/возвращает имя объекта ChartObject
Placement	Variant	Устанавливает/возвращает способ привязки листа ChartObject к листу (например, свободное расположение, перемещение вместе с ячейкой и т.д.) Для установки этого свойства можно воспользоваться константами XlPlacement
PrintObject	Boolean	Устанавливает/возвращает необходимость печати встроенной диаграммы при печати родительского листа
ProtectChartObject	Boolean	Устанавливает/возвращает возможность изменения размера, перемещения и удаления встроенной диаграммы с родительского листа
RoundedCorners	Boolean	Устанавливает/возвращает необходимость закругления углов встроенной диаграммы (True) или отображения правильных углов диаграммы (False)
Shadow	Boolean	Устанавливает/возвращает необходимость отображения тени вокруг встроенной диаграммы
ShapeRange	ShapeRange	Только чтение. Возвращает объект ChartObject в виде объекта Shape
Top	Double	Устанавливает/возвращает расстояние от верхнего края объекта ChartObject до верхнего края родительского листа
TopLeftCell	Range	Только чтение. Возвращает диапазон одной ячейки, расположенной под верхним левым углом объекта ChartObject

Имя	*Тип возвращаемого значения*	*Описание*
Visible	Boolean	Устанавливает/возвращает необходимость отображения объекта ChartObject
Width	Double	Устанавливает/возвращает ширину встроенной диаграммы
ZOrder	Long	Только чтение. Возвращает положение встроенной диаграммы относительно других объектов на листе. Кроме этого, значение свойства ZOrder совпадает с положением объекта ChartObject в родительской коллекции

Методы объекта ChartObject

Имя	*Тип возвращаемого значения*	*Параметры*	*Описание*
Activate	Variant		Делает встроенную диаграмму активной
BringToFront	Variant		Размещает встроенную диаграмму поверх всех остальных объектов на листе. Меняет значение свойства ZOrder
Copy	Variant		Копирует встроенную диаграмму в буфер обмена
CopyPicture	Variant	Appearance As XlPicture-Appearance, Format As XlCopyPicture-Format	Копирует изображение объекта Chart в буфер обмена. Параметр Appearance позволяет выбрать экранное представление или представление для печати. Параметр Format позволяет указать тип изображения, которое будет скопировано в буфер обмена. Параметр Size используется вместе с листами диаграмм для указания размера изображения
Cut	Variant		Вырезает встроенную диаграмму в буфер обмена
Delete	Variant		Удаляет встроенную диаграмму с листа
Duplicate			Дублирует встроенную диаграмму и размещает дубликат на том же родительском листе
Select	Variant	[Replace]	Устанавливает фокус на встроенную диаграмму
SendToBack	Variant		Отправляет встроенную диаграмму на задний фон за всеми объектами на листе

Пример: объект ChartObject и коллекция ChartObjects

В этом примере создаются изображения .jpg для всех встроенных диаграмм на активном листе.

```
Sub ExportChartObjects()
   Dim oCO As ChartObject
   For Each oCO In ActiveSheet.ChartObjects
       ' Экспортировать диаграмму в виде изображения jpg.
       ' Назвать изображение в соответствии с именем
       ' встроенной диаграммы
      oCO.Chart.Export "c:\" & oCO.Name & ".jpg", "jpg"
   Next
End Sub
```

Объект ChartTitle

Объект ChartTitle содержит весь текст и параметры форматирования, связанные с заголовком диаграммы. Для объекта ChartTitle родительским является объект Chart. Обычно этот объект используется совместно со свойством HasTitle родительского объекта Chart.

Общие свойства объекта ChartTitle

Определения свойств Application, Creator и Parent приводились в начале приложения.

Свойства объекта ChartTitle

Имя	*Тип возвращаемого значения*	*Описание*
AutoScaleFont	Variant	Устанавливает/возвращает необходимость автоматического изменения размера шрифта при изменении размеров родительской диаграммы
Border	Border	Только чтение. Возвращает свойства границы вокруг выделенного заголовка диаграммы
Caption	String	Устанавливает/возвращает текст заголовка диаграммы
Characters	Characters	Только чтение. Параметры: [Start], [Length]. Возвращает объект, содержащий все символы заголовка диаграммы. Этот метод позволяет управлять символами заголовка по отдельности
Fill	ChartFillFormat	Только чтение. Возвращает объект, содержащий все параметры форматирования заголовка диаграммы
Font	Font	Только чтение. Возвращает объект, содержащий параметры форматирования шрифта заголовка диаграммы
HorizontalAlignment	Variant	Устанавливает/возвращает параметры горизонтального выравнивания заголовка диаграммы. Для установки этого свойства можно воспользоваться константами xlAlign
Interior	Interior	Только чтение. Возвращает объект, содержащий параметры форматирования области текста в заголовке диаграммы
Left	Double	Устанавливает/возвращает расстояние от левого края заголовка диаграммы до левого края диаграммы
Name	String	Только чтение. Возвращает имя объекта заголовка диаграммы

Имя	*Тип возвращаемого значения*	*Описание*
`Orientation`	`Variant`	Устанавливает/возвращает угол текста в заголовке диаграммы. Значение свойства указывается в градусах в диапазоне от -90 до 90. Кроме этого, для установки свойства можно воспользоваться одной из констант `XlOrientation`
`ReadingOrder`	`Long`	Устанавливает/возвращает направление чтения текста (слева направо или справа налево). Свойство имеет смысл только для некоторых языков
`Shadow`	`Boolean`	Устанавливает/возвращает необходимость отображения эффекта тени для заголовка диаграммы
`Text`	`String`	Устанавливает/возвращает текст заголовка диаграммы
`Top`	`Double`	Устанавливает/возвращает расстояние от верхнего края заголовка диаграммы до верхнего края диаграммы
`Vertical-Alignment`	`Variant`	Устанавливает/возвращает параметры вертикального выравнивания заголовка диаграммы. Для установки этого свойства можно воспользоваться константами `xlVAlign`

Методы объекта ChartTitle

Имя	*Тип возвращаемого значения*	*Параметры*	*Описание*
`Delete`	`Variant`		Удаляет заголовок из диаграммы
`Select`	`Variant`		Выделяет заголовок диаграммы

Пример: объект ChartTitle

В этом примере в диаграмму добавляется заголовок и выполняется его форматирование.

```
Sub AddAndFormatChartTitle()
   Dim oCT As ChartTitle
   ' Удостовериться, что на диаграмме присутствует заголовок
   Charts(1).HasTitle = True
   ' Получить объект ChartTitle
   Set oCT = Charts(1).ChartTitle
   ' Отформатировать заголовок диаграммы
   With oCT
      .Caption = "Hello World"
      .Font.Name = "Times New Roman"
      .Font.Size = 16
      .Characters(1, 1).Font.Color = RGB(255, 0, 0)
      .Characters(7, 1).Font.Color = RGB(255, 0, 0)
      .Border.LineStyle = xlContinuous
      .Border.Weight = xlThin
      .Shadow = True
   End With
End Sub
```

Объект ColorFormat

Объект `ColorFormat` описывает один цвет, который используется родительским объектом. Родительскими объектами могут быть `FillFormat`, `LineFormat`, `ShadowFormat` и `ThreeDFormat`.

Общие свойства объекта ColorFormat

Определения свойств `Application`, `Creator` и `Parent` приводились в начале приложения.

Свойства объекта ColorFormat

Имя	*Тип возвращаемого значения*	*Описание*
`RGB`	`Long`	Только чтение. Возвращает значения красного, синего и зеленого компонентов текущего цвета
`SchemeColor`	`Integer`	Устанавливает/возвращает цвет объекта `ColorFormat` в виде номера в текущей цветовой схеме
`TintAndShade`	`Single`	Устанавливает/возвращает значение, позволяющее осветить или затемнить цвет указанной геометрической фигуры. Значение свойства находится в диапазоне от -1 (самый темный) до 1 (самый светлый). Значение 0 соответствует нейтральной подсветке
`Type`	`MsoColorType`	Только чтение. Возвращает тип цвета: RGB, смешанный или полученный из палитры

Пример: объект ColorFormat

Устанавливает свойство `ForeColor` эффекта заливки геометрической фигуры.

```
Sub FormatShapeColour()
   Dim oShp As Shape
   Dim oCF As ColorFormat
   Set oShp = ActiveSheet.Shapes(1)
   Set oCF = oShp.Fill.ForeColor
   oCF.SchemeColor = 53
End Sub
```

Объект Comment и коллекция Comments

В коллекции `Comments` хранятся все примечания ячеек из родительского объекта `Range`. Каждый объект `Comment` содержит примечание для одной ячейки.

Общие свойства объекта Comment

Определения свойств `Application`, `Creator` и `Parent` приводились в начале приложения.

Свойства объекта Comment

Имя	*Тип возвращаемого значения*	*Описание*
`Author`	`String`	Только чтение. Возвращает имя автора примечания
`Shape`	`Shape`	Только чтение. Возвращает поле примечания в виде объекта `Shape`. Этот объект поддерживает управление параметрами поля примечания
`Visible`	`Boolean`	Устанавливает/возвращает необходимость постоянного отображения примечания (`True`) или отображения примечания только при наведении указателя мыши

Методы объекта Comment

Имя	*Тип возвращаемого значения*	*Параметры*	*Описание*
Delete			Удаляет примечание из ячейки
Next	Comment		Возвращает следующее примечание ячейки из родительской коллекции
Previous	Comment		Возвращает предыдущее примечание ячейки из родительской коллекции
Text	String	[Text], [Start], [Overwrite]	Устанавливает текст примечания. Параметр Text содержит текст примечания. Параметр Start позволяет определить точку вставки в существующее примечание. Для перезаписи существующего текста параметр Overwrite устанавливается в значение True

Пример: объект Comment и коллекция Comments

В этом примере удаляется имя пользователя, добавленное Excel в начало примечания. После этого выполняется форматирование примечания для обеспечения большего удобства при чтении.

```
Sub FormatComments()
   Dim oComment As Comment, i As Integer
   ' Перебрать все примечания в пределах листа
   For Each oComment In ActiveSheet.Comments
      ' Работаем с текстом примечания
      With oComment.Shape.TextFrame.Characters

         ' Найти и удалить имя пользователя, добавленное Excel
         i = InStr(1, .Text, ":" & vbLf)
         If i > 0 Then
            .Text = Mid(.Text, i + 2)
         End If

         ' Увеличить размер шрифта
         With .Font
            .Name = "Arial"
            .Size = 10
            .Bold = False
         End With
      End With

      ' Включить автоматическое изменение размера текстового поля
      oComment.Shape.TextFrame.AutoSize = True
   Next
End Sub
```

Объект ConnectorFormat

Этот объект представляет соединительную линию между геометрическими фигурами. При перемещении любой из геометрических фигур линия автоматически меняет конфигурацию для сохранения визуальной связи между фигурами. Для объекта ConnectorFormat родительским является объект Shape.

Общие свойства ConnectorFormat

Определения свойств Application, Creator и Parent приводились в начале приложения.

Свойства объекта ConnectorFormat

Имя	*Тип возвращаемого значения*	*Описание*
BeginConnected	MsoTriState	Только чтение. Возвращает присутствие геометрической формы в начале соединительной линии. Применяется вместе со свойством BeginConnectedShape
BeginConnectedShape	Shape	Только чтение. Возвращает геометрическую форму, которая связана с началом соединительной линии. Применяется вместе со свойством BeginConnected
BeginConnectionSite	Long	Только чтение. Возвращает точку подключения на геометрической фигуре, с которой связано начало соединительной линии. Используется вместе со свойством BeginConnected
EndConnected	MsoTriState	Только чтение. Возвращает присутствие геометрической формы в конце соединительной линии. Используется вместе со свойством EndConnectedShape
EndConnectedShape	Shape	Только чтение. Возвращает геометрическую фигуру, которая связана с началом соединительной линии. Используется вместе со свойством EndConnected
EndConnectionSite	Long	Только чтение. Возвращает точку подключения на геометрической фигуре, с которой связан конец соединительной линии. Используется вместе со свойством EndConnected
Type	MsoConnectorType	Устанавливает/возвращает тип используемой соединительной линии (например, msoConnectorStraight, msoConnectorCurve)

Методы объекта ConnectorFormat

Имя	*Тип возвращаемого значения*	*Параметры*	*Описание*
BeginConnect		ConnectedShape As Shape, ConnectionSite As Long	Связывает начало соединительной линии с геометрической фигурой, указанной в качестве значения параметра ConnectedShape. Точка подключения указывается в качестве значения параметра ConnectionSite
Disconnect			Отключает геометрическую фигуру, которая находится в начале соединительной линии. Этот метод не перемещает соединительную линию
EndConnect		ConnectedShape As Shape, ConnectionSite As Long	Связывает конец соединительной линии с геометрической фигурой, указанной в качестве значения параметра ConnectedShape. Точка подключения указывается в качестве значения параметра ConnectionSite

Имя	*Тип возвращаемого значения*	*Параметры*	*Описание*
`End-Disconnect`			Отключает геометрическую фигуру, которая находится в конце соединительной линии. Этот метод не перемещает соединительную линию

Пример: объект ConnectorFormat

В этом примере все соединительные линии форматируются в виде кривых.

```
Sub FormatConnectors()
   Dim oShp As Shape
   Dim oCF As ConnectorFormat
   ' Перебрать все объекты Shape в пределах листа
   For Each oShp In ActiveSheet.Shapes
      ' Это объект Connector?
      If oShp.Connector Then

         ' Если да, получить объект ConnectorFormat
         Set oCF = oShp.ConnectorFormat

         ' Если соединительная линия подключена с обоих сторон,
         ' превратить ее в кривую.
         With oCF
            If .BeginConnected And .EndConnected Then
               .Type = msoConnectorCurve
            End If
         End With
      End If
   Next
End Sub
```

Объект ControlFormat

Объект `ControlFormat` содержит свойства и методы, позволяющие управлять такими элементами управления, как поля ввода и списки. Родительским для этого объекта всегда является объект `Shape`.

Общие свойства объекта ControlFormat

Определения свойств `Application`, `Creator` и `Parent` приводились в начале приложения.

Свойства объекта ControlFormat

Имя	*Тип возвращаемого значения*	*Описание*
`DropDown-Lines`	`Long`	Устанавливает/возвращает количество строк, отображаемых в раскрывающейся части комбинированного списка. Используется только с комбинированными списками
`Enabled`	`Boolean`	Устанавливает/возвращает состояние элемента управления (включен/выключен)
`LargeChange`	`Long`	Устанавливает/возвращает значение полосы прокрутки, которое добавляется или отнимается при каждом щелчке пользователя на внутренней области полосы прокрутки. Используется только с полосой прокрутки

Имя	*Тип возвращаемого значения*	*Описание*
LinkedCell	String	Устанавливает/возвращает диапазон, в который копируются результаты работы элемента управления
ListCount	Long	Только чтение. Возвращает количество элементов в поле списка или в комбинированном списке. Применяется только с полосой прокрутки
ListFill-Range	String	Устанавливает/возвращает диапазон, в котором хранятся элементы для поля списка или комбинированного списка. Используется только с комбинированным списком или полем списка
ListIndex	Long	Устанавливает/возвращает элемент, который в данный момент выделен в поле списка или в комбинированном списке. Используется только с комбинированным списком или полем списка
LockedText	Boolean	Устанавливает/возвращает возможность модификации текста в элементе управления при блокировке книги
Max	Long	Устанавливает/возвращает максимальное значение, которое может принимать полоса прокрутки или поле счетчика. Применяется только с полосой прокрутки или счетчиком
Min	Long	Устанавливает/возвращает минимальное значение, которое может принимать полоса прокрутки или поле счетчика. Используется только с полосой прокрутки или счетчиком
MultiSelect	Long	Устанавливает/возвращает реакцию списка на пользовательское выделение. Это свойство может устанавливаться в значение xlNone (выделяется только один элемент списка), xlSimple (в выделение добавляется каждый элемент, на котором щелкает пользователь) или xlExtended (для выделения нескольких элементов списка пользователь должен удерживать клавишу <Ctrl>). Используется только с полями списков
PrintObject	Boolean	Устанавливает/возвращает необходимость печати элементов управления при печати листа
SmallChange	Long	Устанавливает/возвращает значение, которое добавляется или отнимается от значения полосы прокрутки, когда пользователь щелкает на стрелке полосы прокрутки. Применяется только с полосами прокрутки
Value	Long	Устанавливает/возвращает значение элемента управления

Методы ControlFormat

Имя	*Тип возвращаемого значения*	*Параметры*	*Описание*
AddItem		Text As String, [Index]	Добавляет значение параметра Text в поле списка или в комбинированный список. Этот метод может использоваться только с полями списков или комбинированными списками

Имя	*Тип возвращаемого значения*	*Параметры*	*Описание*
`List`	`Variant`	`[Index]`	Устанавливает/возвращает массив списка строк, связанный с комбинированным списком или полем списка. Кроме этого, метод устанавливает/возвращает отдельные элементы списка, если указать значение параметра `Index`. Используется только с полями списков и с комбинированными списками
`Remove-AllItems`			Удаляет все элементы из поля списка или комбинированного списка. Используется только с полями списков и с комбинированными списками
`RemoveItem`		`Index As Long, [Count]`	Удаляет элементы, на которые указывает значение параметра `Index`. Используется только с полями списков и с комбинированными списками

Пример: объект ControlFormat

В этом примере сбрасываются все поля списка, раскрывающиеся списки, полосы прокрутки, счетчики и флажки на листе.

```
Sub ResetFormControls()
   Dim oShp As Shape
   Dim oCF As ControlFormat
   ' Перебрать все геометрические фигуры на листе
   For Each oShp In ActiveSheet.Shapes
      ' Это элемент управления Формы (Forms)?
      If oShp.Type = msoFormControl Then

         ' Да, получить объект ControlFormat
         Set oCF = oShp.ControlFormat

         ' Сбросить элемент управления соответствующим образом
         Select Case oShp.FormControlType
            Case xlListBox, xlDropDown
               oCF.RemoveAllItems

            Case xlSpinner, xlScrollBar
               oCF.Value = oCF.Min

            Case xlCheckBox
               oCF.Value = xlOff

         End Select
      End If
   Next
End Sub
```

Объект Corners

Этот объект представляет углы трехмерной диаграммы. Для него родительским объектом является объект `Chart`. Родительская диаграмма должна быть трехмерной. Доступ к отдельным углам невозможен.

Общие свойства объекта Corners

Определения свойств Application, Creator и Parent приводились в начале приложения.

Свойства объекта Corners

Имя	*Тип возвращаемого значения*	*Описание*
Name	String	Только чтение. Возвращает имя объекта Corners — обычно это "Corners"

Методы объекта Corners

Имя	*Тип возвращаемого значения*	*Параметры*	*Описание*
Select	Variant		Выделяет углы диаграммы

Пример: объект Corners

Пример отсутствует — существует только метод для выделения углов, что не несет практической пользы.

Объект CubeField и коллекция CubeFields

В коллекции CubeFields хранятся все поля отчета сводной таблицы, основанные на кубе OLAP. Каждый объект CubeField представляет измерение или иерархию полей из куба OLAP. Родительским объектом для коллекции CubeFields является объект PivotTable.

Кроме типичных атрибутов, коллекция CubeFields предоставляет свойство Count. Свойство Count возвращает количество объектов в коллекции.

Методы коллекции CubeFields

Имя	*Тип возвращаемого значения*	*Параметры*	*Описание*
AddSet	CubeField	Name As String, Caption As String	Добавляет новый объект CubeField в коллекцию CubeFields

Общие свойства объекта CubeField

Определения свойств Application, Creator и Parent приводились в начале приложения.

Свойства объекта CubeField

Имя	*Тип возвращаемого значения*	*Описание*
Caption	String	Только чтение. Возвращает текстовую метку, которая используется для поля куба
CubeField-Type	XlCubeField-Type	Только чтение. Возвращает значение, определяющее, является ли поле куба полем иерархии (xlHierarchy) или полем измерения (xlMeasure)

Имя	*Тип возвращаемого значения*	*Описание*
DragToColumn	Boolean	Устанавливает/возвращает возможность перетаскивания поля в позицию столбца. Для полей измерений это свойство равно значению False
DragToData	Boolean	Устанавливает/возвращает возможность перетаскивания поля в позицию данных
DragToHide	Boolean	Устанавливает/возвращает возможность перетаскивания поля из отчета сводной таблицы (сокрытия поля)
DragToPage	Boolean	Устанавливает/возвращает возможность перетаскивания поля в позицию страницы. Для полей измерений это свойство равно значению False
DragToRow	Boolean	Устанавливает/возвращает возможность перетаскивания поля в позицию строки. Для полей измерений это поле равно значению False
Enable-Multiple-PageItems	Boolean	Устанавливает/возвращает возможность выделения нескольких элементов в области полей страницы для сводных таблиц OLAP
HasMember-Properties	Boolean	Только чтение. Возвращает значение True, если выбраны свойства-члены, которые должны отображаться для поля куба
HiddenLevels	Long	Устанавливает/возвращает скрываемые верхние уровни иерархии поля куба. Перед установкой свойства в значение, превышающее 0, нужно установить свойство в значение 0 (при этом будут показаны все уровни, после чего часть уровней будет скрыта)
LayoutForm	XlLayout-FormType	Устанавливает/возвращает способ отображения выбранных элементов сводной таблицы
Layout-Subtotal-Location	XlSubtotal-LocationType	Устанавливает/возвращает положение итогов полей сводной таблицы относительно выбранных полей
Name	String	Только чтение. Возвращает имя объекта
Orientation	XlPivotField-Orientation	Устанавливает/возвращает расположение поля в отчете сводной таблицы
PivotFields	PivotFields	Только чтение. Возвращает коллекцию PivotFields
Position	Long	Устанавливает/возвращает номер позиции поля иерархии среди всех полей той же ориентации
ShowInField-List	Boolean	Устанавливает/возвращает необходимость отображения объекта CubeField в списке полей
Treeview-Control	TreeviewControl	Только чтение. Возвращает объект, позволяющий управлять кубом в отчете сводной таблицы OLAP
Value	String	Только чтение. Возвращает имя поля

Методы объекта CubeField

Имя	*Тип возвращаемого значения*	*Параметры*	*Описание*
AddMember-PropertyField		Property As String, [Property Order]	Добавляет поле свойства в отображение поля куба. Обратите внимание, что выбранное поле свойства будет недоступным для просмотра, если в представлении сводной таблицы отсутствуют поля
Delete			Удаляет объект

Объект CustomProperty и коллекция CustomProperties

Данный объект позволяет сохранять информацию внутри листа или внутри смарт-тега. Эта информация может использоваться в качестве метаданных для генерации кода XML. Также она доступна любой подпрограмме, работающей с листом или смарт-тегом.

Более важным аспектом этого объекта с точки зрения разработчика является возможность хранения информации, специфичной для листа или группы листов. В таком случае подпрограмма может обращаться к объекту `CustomProperty`, анализировать полученную информацию и принимать решение о дальнейшей обработке листа. Раньше для хранения информации о листе многие разработчики использовали имена диапазонов на уровне листа, которые хранились только в пределах этого листа, что позволяло использовать такие же имена в других листах для хранения других данных.

Например, каждый лист в пределах книги содержит десяток бюджетных листов и три листа с отчетами, в которых могут присутствовать одинаковые имена диапазонов `IsBudget`. Во всех бюджетных листах в этом имени хранится значение `True`. На листах отчетов в этом имени хранится значение `False`. Если подпрограмме необходимо перебрать листы и применить различное форматирование в зависимости от типа листа, подпрограмма может проверить значение имени диапазона и определить, обрабатывается ли бюджетный лист.

Новый объект `CustomProperty` значительно упрощает хранение такой информации (или любой другой информации), позволяя отказаться от использования диапазонов уровня листа, хранения информации на скрытом листе или применения системного реестра.

Коллекция `CustomProperties` предоставляет доступ к объектам `CustomProperty` для листов и смарт-тегов. В объекте `CustomProperty` может храниться информация как для листа, так и для смарт-тега. Эти объекты напоминают объекты `DocumentProperty` из объектной модели OfficeXP, но они хранятся в пределах листа или смарт-тега, а не для всего документа в целом.

Общие свойства коллекции CustomProperties

Определения свойств `Application`, `Creator` и `Parent` приводились в начале приложения.

Свойства коллекции CustomProperties

Имя	*Тип возвращаемого значения*	*Описание*
Count	Long	Только чтение. Возвращает количество объектов в пределах коллекции
Item	CustomProperty	Только чтение. `Index As Variant`. Возвращает один объект из коллекции

Методы коллекции CustomProperties

Имя	Тип возвращаемого значения	Параметры	Описание
Add	CustomProperty	Name As String, Value As Variant	Добавляет информацию в дополнительное свойство

Общие свойства объекта CustomProperty

Определения свойств Application, Creator и Parent приводились в начале приложения.

Свойства объекта CustomProperty

Имя	Тип возвращаемого значения	Описание
Name	String	Устанавливает/возвращает имя объекта
Value	Variant	Устанавливает/возвращает стиль линии границ (например, xlDash). Для установки значения свойства можно воспользоваться константами xlLineStyle. Свойство является аналогом свойства LineStyle

Методы объекта CustomProperty

Имя	Тип возвращаемого значения	Параметры	Описание
Delete			Удаляет объект

Пример: объект CustomProperty

Эта подпрограмма последовательно перебирает листы в пределах книги и создает объект CustomProperty, который называется IsBudget. Значение объекта IsBudget зависит от того, содержит ли лист фразу “Бюджетный анализ”. После этого выводится результат работы подпрограммы.

```
Sub CreateCustomProperties()
    Dim bBudget As Boolean
    Dim lRow As Long
    Dim oCustomProp As CustomProperty
    Dim rng As Range, wks As Worksheet

    ' Отключить обновление экрана и очистить форматы поиска
    With Application
        .FindFormat.Clear
        .ScreenUpdating = False
    End With

    ' Очистить лист, на котором будет хранится список
    ' объектов CustomProperty
    wksCustomProperties.UsedRange.Offset(1, 0).ClearContents

    ' Инициализировать счетчик строк
    lRow = 2    ' В первой строке находятся заголовки столбцов

    ' Перебрать листы в пределах книги
    For Each wks In ThisWorkbook.Worksheets

        ' Подавить вывод сообщений об ошибках поиска
```

```
        ' ячеек и ошибках поиска объектов CustomProperty
        On Error Resume Next
            bBudget = False

            bBudget = _
                (Len(wks.UsedRange.Find(What:="Budget
↳ Analysis").Address) > 0)

            ' К сожалению, нельзя ссылаться на объект CustomProperty
            ' по имени. Только по номеру.
            Set oCustomProp = wks.CustomProperties(1)
        On Error GoTo 0

        ' Если объект CustomProperty существует, удалить его
        ' и добавить заново
        If Not oCustomProp Is Nothing Then oCustomProp.Delete

        ' Обратите внимание, что значение bBudget заключено в
        ' двойные кавычки.
        ' Если этого не сделать, True будет сохранено как -1,
        ' а False -- как 0 (соответствующие числовые значения)
        Set oCustomProp = wks.CustomProperties.Add(Name:=
↳"IsBudget", Value:="" & bBudget & "")

        ' Перечислить параметры объектов CustomProperty на листе.
        With wksCustomProperties
            ' Вызов Parent.Name возвращает имя объекта,
            ' в котором хранится объект CustomProperty.
            ' В данном случае это имя листа.
            .Cells(lRow, 1).Value = oCustomProp.Parent.Name
            .Cells(lRow, 2).Value = oCustomProp.Name
            .Cells(lRow, 3).Value = oCustomProp.Value
        End With

        ' Сместиться на одну строку вниз
        lRow = lRow + 1

    Next wks

End Sub
```

Объект CustomView и коллекция CustomViews

В коллекции CustomViews хранится список дополнительных представлений, связанных с книгой. В каждом объекте CustomView хранятся атрибуты дополнительного представления книги. В дополнительном представлении хранятся такие параметры, как размер и положение окна, ширина столбцов, скрытые столбцы, а также параметры печати книги. Для коллекции CustomViews родительским объектом является объект Workbook.

Кроме типичных атрибутов, коллекция CustomViews предоставляет два дополнительных свойства. Свойство Count возвращает количество объектов CustomView в пределах коллекции. Метод Add позволяет добавить представление в коллекцию CustomViews. В качестве значения параметра ViewName метод Add принимает имя представления. При необходимости в представление можно добавить параметры печати (PrintSettings) и скрытые строки и столбцы (RowColSettings).

Общие свойства объекта CustomView

Определения свойств Application, Creator и Parent приводились в начале приложения.

Свойства объекта CustomView

Имя	*Тип возвращаемого значения*	*Описание*
Name	String	Только чтение. Возвращает имя дополнительного представления
PrintSettings	Boolean	Только чтение. Возвращает необходимость включения представление параметров печати
RowColSettings	Boolean	Только чтение. Возвращает необходимость включения в представление скрытых строк и столбцов

Методы объекта CustomView

Имя	*Тип возвращаемого значения*	*Параметры*	*Описание*
Delete			Удаляет дополнительное представление
Show			Отображает дополнительное представление и связанные с ним параметры

Пример: объект CustomView и коллекция CustomViews

Подпрограмма последовательно выводит дополнительные представления книги, выдерживая паузу в 2 секунды.

```
Sub ShowCustomView()
   Dim oCV As CustomView
   ' Перебрать все представления, содержащие информацию о строках
   ' и столбцах.
   For Each oCV In ActiveWorkbook.CustomViews
      If oCV.RowColSettings Then
         oCV.Show
      End If

      ' Приостановить выполнение приложения на 2 секунды
      Application.Wait Now + TimeValue("00:00:02")
   Next
End Sub
```

Объект DataLabel и коллекция DataLabels

В коллекции DataLabels хранятся все метки отдельных точек и линий трендов для ряда данных. В каждом ряду присутствует только одна коллекция DataLabels. Для коллекции DataLabels родительским является объект Series. Каждый объект DataLabel представляет одну метку данных для линии тренда или точки данных. Коллекция DataLabels используется вместе свойством HasDataLabels родительского объекта Series.

Коллекция DataLabels предоставляет несколько свойств и методов кроме типичных атрибутов коллекции. Эти методы и свойства перечислены в следующей таблице.

Свойства и методы коллекции DataLabels

Имя	*Тип возвращаемого значения*	*Описание*
AutoScaleFont	Variant	Устанавливает/возвращает необходимость автоматического изменения размера шрифта при изменении размера родительской диаграммы
AutoText	Boolean	Устанавливает/возвращает необходимость автоматической генерации текста меток данных
Border	Border	Только чтение. Возвращает свойства границы вокруг коллекции меток данных
Count	Long	Только чтение. Возвращает количество меток данных в коллекции
Fill	ChartFillFormat	Только чтение. Возвращает объект, содержащий параметры форматирования заливки меток данных в коллекции
Font	Font	Только чтение. Возвращает объект, содержащий параметры шрифта для меток данных в коллекции
HorizontalAlignment	Variant	Устанавливает/возвращает параметры горизонтального выравнивания меток данных. Для установки свойства можно воспользоваться константами xlAlign
Interior	Interior	Только чтение. Возвращает объект, содержащий параметры форматирования внутренней области меток данных в коллекции (например, внутренний цвет)
Name	String	Только чтение. Возвращает имя коллекции
NumberFormat	String	Устанавливает/возвращает формат чисел, который используется, если метки данных являются числами или датами
NumberFormatLinked	Boolean	Устанавливает/возвращает необходимость применения того же формата чисел, который используется в ячейках с данными для меток данных
NumberFormatLocal	Variant	Устанавливает/возвращает имя формата чисел, который используется в метках данных. Имя указывается на языке пользовательского интерфейса
Orientation	Variant	Устанавливает/возвращает угол, под которым расположен текст меток данных. Значение свойства можно указывать в градусах (в пределах от -90 до 90) или воспользоваться одной из констант XlOrientation
Position	XlDataLabelPosition	Устанавливает/возвращает расположение меток данных относительно точек или линий тренда
ReadingOrder	Long	Устанавливает/возвращает направление текста (слева направо или справа налево). Это свойство может использоваться с некоторыми языками пользовательского интерфейса
Separator	Variant	Устанавливает/возвращает разделитель, который используется для меток данных на диаграммах
Shadow	Boolean	Устанавливает/возвращает необходимость отображения тени вокруг меток данных
ShowBubbleSize	Boolean	Устанавливает/возвращает необходимость отображения размера пузырька для меток данных на диаграмме

Имя	*Тип возвращаемого значения*	*Описание*
ShowCategoryName	Boolean	Устанавливает/возвращает необходимость отображения имени категории для меток данных на диаграмме
ShowLegendKey	Boolean	Устанавливает/возвращает необходимость отображения ключа из легенды (обычно это один из цветов) вместе с меткой данных
ShowPercentage	Boolean	Устанавливает/возвращает необходимость отображения процентного значения вместе с меткой данных на диаграмме
ShowSeriesName	Boolean	Устанавливает/возвращает необходимость отображения имени ряда
ShowValue	Boolean	Устанавливает/возвращает необходимость отображения значений меток данных конкретной диаграммы
Type	Variant	Устанавливает/возвращает тип отображаемых меток данных для коллекции (например, метки, проценты, значения)
VerticalAlignment	Variant	Устанавливает/возвращает параметры вертикального выравнивания меток данных. Для установки этого свойства можно воспользоваться константами xlVAlign
Delete	Variant	Метод. Удаляет метки данных
Select	Variant	Метод. Выделяет метки данных на диаграмме

Общие свойства объекта DataLabel

Определения свойств Application, Creator и Parent приводились в начале приложения.

Свойства объекта DataLabel

Имя	*Тип возвращаемого значения*	*Описание*
AutoScaleFont	Variant	Устанавливает/возвращает необходимость автоматического изменения размера шрифта при изменении размера родительской диаграммы
AutoText	Boolean	Устанавливает/возвращает необходимость автоматической генерации текста меток данных средствами Excel
Border	Border	Только чтение. Возвращает свойства границы вокруг метки данных
Caption	String	Устанавливает/возвращает текст метки данных
Characters	Characters	Только чтение. Параметры: [Start], [Length]. Возвращает объект, который представляет диапазон символов текста
Fill	ChartFillFormat	Только чтение. Возвращает объект, содержащий параметры форматирования заливки для меток данных
Font	Font	Только чтение. Возвращает объект, содержащий параметры шрифта для метки данных
HorizontalAlignment	Variant	Устанавливает/возвращает параметры горизонтального выравнивания метки данных. Для установки этого свойства используются константы xlAlign

Имя	*Тип возвращаемого значения*	*Описание*
`Interior`	`Interior`	Только чтение. Возвращает объект, содержащий параметры форматирования внутренней области метки данных (например, цвет внутренней области)
`Left`	`Double`	Устанавливает/возвращает расстояние от левого края метки данных до левого края родительской диаграммы
`Name`	`String`	Только чтение. Возвращает имя метки данных
`NumberFormat`	`String`	Устанавливает/возвращает формат чисел, который используется для меток данных, являющихся числом или датой
`Number-Format-Linked`	`Boolean`	Устанавливает/возвращает необходимость использования одинакового формата чисел для меток данных и ячеек, содержащих данные для диаграммы
`Number-FormatLocal`	`Variant`	Устанавливает/возвращает имя формата чисел для меток данных на языке пользовательского интерфейса
`Orientation`	`Variant`	Устанавливает/возвращает угол расположения текста для метки данных. Значение этого свойства можно указывать в градусах (в пределах от -90 до 90). Кроме этого, можно воспользоваться константами `XlOrientation`
`Position`	`XlDataLabel-Position`	Устанавливает/возвращает расположение метки данных относительно точек или линий тренда
`ReadingOrder`	`Long`	Устанавливает/возвращает направление текста (слева направо или справа налево). Это свойство может использоваться с некоторыми языками пользовательского интерфейса
`Separator`	`Variant`	Устанавливает/возвращает разделитель, который используется с метками данных на диаграмме
`Shadow`	`Boolean`	Устанавливает/возвращает необходимость отображения тени для меток данных
`ShowBubble-Size`	`Boolean`	Устанавливает/возвращает необходимость отображения размера пузырька для меток данных на диаграмме
`Show-Category-Name`	`Boolean`	Устанавливает/возвращает необходимость отображения имени категории для меток данных на диаграмме
`ShowLegend-Key`	`Boolean`	Устанавливает/возвращает необходимость отображения ключа из легенды (обычно это один из цветов) вместе с меткой данных
`Show-Percentage`	`Boolean`	Устанавливает/возвращает необходимость отображения процентного значения вместе с меткой данных на диаграмме
`ShowSeries-Name`	`Boolean`	Устанавливает/возвращает необходимость отображения имени последовательности
`ShowValue`	`Boolean`	Устанавливает/возвращает необходимость отображения значений меток данных конкретной диаграммы
`Text`	`String`	Устанавливает/возвращает текст метки данных
`Top`	`Double`	Устанавливает/возвращает расстояние от верхнего края метки данных до верхнего края родительской диаграммы
`Type`	`Variant`	Устанавливает/возвращает тип отображаемых меток данных (например, метки, проценты, значения)
`Vertical-Alignment`	`Variant`	Устанавливает/возвращает параметры вертикального выравнивания меток данных. Для установки этого свойства используются константы `xlVAlign`

Методы объекта DataLabel

Имя	Тип возвращаемого значения	Параметры	Описание
Delete	Variant		Удаляет метку данных
Select	Variant		Выделяет метку данных на диаграмме

Пример: объект DataLabel и коллекция DataLabels

В данном примере метки данных добавляются ко всем точкам диаграммы. Для этого используется столбец слева от диапазона значений для оси X.

```
Sub AddDataLabels()
    Dim oSer As Series
    Dim vaSplits As Variant
    Dim oXRng As Range
    Dim oLblRng As Range
    Dim oLbl As DataLabel
    ' Перебрать все ряды диаграммы
    For Each oSer In Charts(1).SeriesCollection
       ' Получить формулу ряда и разделить ее на
       ' составляющие (Имя, диапазон X, диапазон Y, порядок)
       vaSplits = Split(oSer.Formula, ",")

       'Получить диапазон X
       Set oXRng = Range(vaSplits(LBound(vaSplits) + 1))

       'Получить столбец слева от диапазона X
       Set oLblRng = oXRng.Offset(0, -1)

       ' Отобразить метки данных для рядов
       oSer.ApplyDataLabels

       ' Перебрать точки
       For i = 1 To oSer.Points.Count

          ' Получить объект DataLabel
          Set oLbl = oSer.Points(i).DataLabel
          ' Получить текст и выравнивание
          With oLbl
             .Caption = oLblRng.Cells(i)
             .Position = xlLabelPositionAbove
       End With
    Next
    Next
End Sub
```

Объект DataTable

В этом объекте хранятся параметры форматирования, связанные с таблицей данных диаграммы. Родительским для этого объекта является объект Chart.

Общие свойства объекта DataTable

Определения свойств Application, Creator и Parent приводились в начале приложения.

Свойства объекта DataTable

Имя	*Тип возвращаемого значения*	*Описание*
AutoScale-Font	Variant	Устанавливает/возвращает необходимость автоматического изменения размера шрифта при изменении размера родительской диаграммы
Border	Border	Только чтение. Возвращает свойства границы вокруг таблицы данных
Font	Font	Только чтение. Возвращает объект, содержащий параметры шрифта для таблицы данных
HasBorder-Horizontal	Boolean	Устанавливает/возвращает необходимость отображения горизонтальных границ ячеек в таблице данных
HasBorder-Outline	Boolean	Устанавливает/возвращает необходимость отображения внешних границ ячеек в таблице данных
HasBorder-Vertical	Boolean	Устанавливает/возвращает необходимость отображения вертикальных границ ячеек в таблице данных
ShowLegend-Key	Boolean	Устанавливает/возвращает необходимость отображения ключа легенды вместе с содержимым таблицы данных

Методы объекта DataTable

Имя	*Тип возвращаемого значения*	*Параметры*	*Описание*
Delete			Удаляет таблицу данных
Select			Выделяет таблицу данных на диаграмме

Пример: объект DataTable

Добавляет таблицу данных в диаграмму и меняет форматирование таким образом, чтобы между значениями отображались вертикальные линии.

```
Sub FormatDataTable()
   Dim oDT As DataTable
   ' Отобразить таблицу данных
   Charts(1).HasDataTable = True
   ' Получить объект DataTable
   Set oDT = Charts(1).DataTable
   ' Отформатировать таблицу данных таким образом,
   ' чтобы отображались только вертикальные линии
   With oDT
       .HasBorderOutline = False
       .HasBorderHorizontal = False
       .HasBorderVertical = True
   End With
End Sub
```

Объект DefaultWebOptions

Поддерживает программный доступ к принятым по умолчанию параметрам диалогового окна Параметры веб-документа (Web Options). Эти параметры определяют поведение Excel при открытии страницы HTML и при сохранении листа в виде страницы HTML.

Общие свойства объекта DefaultWebOptions

Определения свойств Application, Creator и Parent приводились в начале приложения.

Свойства объекта DefaultWebOptions

Имя	*Тип возвращаемого значения*	*Описание*
AllowPNG	Boolean	Устанавливает/возвращает возможность использования формата Portable Network Graphics (PNG) в качестве выходного формата изображений. Этот переносимый формат поддерживает сжатие изображений без потерь, обеспечивая при этом достаточно высокую степень сжатия
AlwaysSave-InDefault-Encoding	Boolean	Устанавливает/возвращает необходимость сохранения Web-документов с использованием принятой по умолчанию кодировки
CheckIf-Office-IsHTMLEditor	Boolean	Устанавливает/возвращает, является ли пакет Office принятым по умолчанию редактором для страниц, созданных средствами Office
Download-Components	Boolean	Устанавливает/возвращает информацию о присутствии компонентов Office на компьютере пользователя, когда пользователь просматривает файлы Excel в Web-обозревателе
Encoding	MsoEncoding	Устанавливает/возвращает тип кодировки, в которой сохраняются документы
FolderSuffix	String	Только чтение. Возвращает суффикс для каталога поддержки, который создается при сохранении документа Excel в виде Web-документа. Это свойство зависит от языка
Fonts	WebPageFonts	Только чтение. Возвращает коллекцию допустимых Web-шрифтов
Load-Pictures	Boolean	Устанавливает/возвращает необходимость загрузки изображений при открытии файла Excel
LocationOf-Components	String	Устанавливает/возвращает адрес URL или путь, по которому доступны Web-компоненты Office, необходимые для просмотра документа в Web-обозревателе
OrganizeIn-Folder	Boolean	Устанавливает/возвращает необходимость сохранения файлов поддержки в отдельной папке
PixelsPer-Inch	Long	Устанавливает/возвращает способ отображения графика плотности и ячеек таблицы на Web-странице
RelyOnCSS	Boolean	Устанавливает/возвращает необходимость использования каскадных листов стилей для форматирования шрифтов
RelyOnVML	Boolean	Устанавливает/возвращает необходимость отказа от генерации изображений при сохранении документа с рисованными объектами. Для создания изображений на лету используется язык Vector Markup Language (VML). Этот язык основан на XML и обеспечивает генерацию векторной графики для Web-страниц
SaveHidden-Data	Boolean	Устанавливает/возвращает необходимость сохранения на Web-странице всех скрытых данных

Имя	*Тип возвращаемого значения*	*Описание*
SaveNewWebPagesAsWebArchives	Boolean	Устанавливает/возвращает возможность сохранения Web-страницы в виде Web-архива
ScreenSize	MsoScreenSize	Устанавливает/возвращает размер экрана целевого монитора
TargetBrowser	MsoTargetBrowser	Устанавливает/возвращает версию обозревателя
UpdateLinksOnSave	Boolean	Устанавливает/возвращает необходимость обновления ссылок при каждом сохранении документа
UseLongFileNames	Boolean	Устанавливает/возвращает необходимость использования длинных имен файлов, если такая возможность существует

Пример: объект DefaultWebOptions

В этом примере показано, как открывать Web-страницу, не загружая изображения.

```
Sub OpenHTMLWithoutPictures()
   Dim bLoadImages As Boolean
   Dim oDWO As DefaultWebOptions
   ' Получить объект DefaultWebOptions
   Set oDWO = Application.DefaultWebOptions
   ' Сохранить  необходимость загрузки изображений
   bLoadImages = oDWO.LoadPictures
   ' Сообщить Excel о том, что изображения загружать не нужно
   oDWO.LoadPictures = False
   ' Открыть Web-страницу без изображений
   Workbooks.Open "http://www.wrox.com"
   ' Восстановить параметры
   oDWO.LoadPictures = bLoadImages
End Sub
```

Объект Diagram

Объект Diagram представляет предопределенную коллекцию геометрических форм, окруженных невидимой границей. Это комбинация добавления геометрических фигур вручную с помощью панели инструментов Рисование (Drawing) и использованием расширенной версии программы Организационная диаграмма, которая предоставлялась в предыдущих версиях Microsoft Office. Внутри каждого объекта Diagram хранятся объекты Node. Каждый объект Node представляет отдельную геометрическую фигуру в пределах диаграммы.

Существует несколько предопределенных объектов Diagram, доступных пользователю: Cycle, Target, Radial, Venn, Pyramid и OrgChart.

Важно обратить внимание, что объект Diagram принадлежит объекту Shape(s), который, в свою очередь, принадлежит объекту Worksheet. Следовательно, для добавления объекта Diagram в существующий лист необходимо использовать метод AddDiagram коллекции Shapes.

```
ActiveSheet.Shapes.AddDiagram(msoDiagramOrgChart, 2, 2, 400, 300)
```

Если результат выполнения предыдущего кода присвоить объектной переменной, код будет возвращать объект Shape. Для добавления геометрических фигур в объект Diagram можно воспользоваться объектом DiagramNode в пределах объекта Shape.

```
ActiveSheet.Shapes(1).DiagramNode.Children.AddNode
```

Для обращения к свойствам и методам самого объекта Diagram (эти свойства и методы показаны ниже) необходимо обращаться к объекту Diagram через объект Shape.

```
ActiveSheet.Shapes(1).Diagram.Nodes(1).TextShape.Fill.BackColor. _
SchemeColor = 17
```

Общие свойства объекта Diagram

Определения свойств Application, Creator и Parent приводились в начале приложения.

Свойства объекта Diagram

Имя	*Тип возвращаемого значения*	*Описание*
AutoFormat	MsoTriState	Устанавливает/возвращает состояние автоматического форматирования
AutoLayout	MsoTriState	Устанавливает/возвращает константу, которая определяет автоматическое размещение узлов и соединительных линий в пределах диаграммы
Nodes	DiagramNodes	Только чтение. Возвращает объект DiagramNodes, который содержит простой список всех узлов из указанной диаграммы
Reverse	MsoTriState	Устанавливает/возвращает необходимость обратного порядка перечисления узлов
Type	MsoDiagramType	Только чтение. Возвращает тип диаграммы

Методы объекта Diagram

Имя	*Тип возвращаемого значения*	*Параметры*	*Описание*
Convert		Type As Mso DiagramType	Выполняет преобразование текущей диаграммы в другую диаграмму

Пример: объекта Diagram

Следующая подпрограмма создает диаграмму, добавляет несколько геометрических фигур (узлов) и выполняет форматирование. Цвет геометрической фигуры и имя шрифта извлекаются из таблицы на листе, что позволяет легко экспериментировать с внешним видом диаграммы.

На момент написания этой книги любая попытка программного добавления текста в узлы объекта Diagram приводила к ошибке.

```
Sub CreateDiagram()

    Const sRANGE_LEVELS As String = "Levels"

    Dim lCount As Long
    Dim oDiagramShape As Shape
    Dim oDiagramNode As DiagramNode
    Dim oDiagramNodeChild As DiagramNode

    ' Удалить существующие геометрические фигуры (кроме
    ' командной кнопки)
    On Error Resume Next
        For Each oDiagramShape In wksDiagrams.Shapes
```

```
            If oDiagramShape.HasDiagram Then oDiagramShape.Delete
        Next oDiagramShape
    On Error GoTo 0

    ' Отключить обновление экрана
    Application.ScreenUpdating = False
    ' Создать объект Diagram
    Set oDiagramShape = wksDiagrams.Shapes.
➥AddDiagram(msoDiagramOrgChart, 2, 2, 300, 250)

    ' Удалить прозрачный фон
    oDiagramShape.Fill.Visible = msoTrue

    ' Создать узел верхнего уровня
    Set oDiagramNode = oDiagramShape.DiagramNode.Children.AddNode

    With oDiagramNode

        ' Выполнить форматирование узла верхнего уровня
    With .Shape
        .AutoShapeType = msoShapeBevel
        .TextFrame.Characters.Font.Name = _
            wksDiagrams.Range(sRANGE_LEVELS).Cells(1, 2).Text
        .Fill.ForeColor.SchemeColor = _
            wksDiagrams.Range(sRANGE_LEVELS).Cells(1, 3).Value
    End With

    ' Создать дочерний узел под узлом верхнего уровня
    Set oDiagramNodeChild = .Children.AddNode

    ' Выполнить форматирование дочернего узла
    With oDiagramNodeChild
        .Shape.TextFrame.Characters.Font.Name = _
            wksDiagrams.Range(sRANGE_LEVELS).Cells(2, 2).Text
        .Shape.Fill.ForeColor.SchemeColor = _
            wksDiagrams.Range(sRANGE_LEVELS).Cells(2, 3).Value
    End With

    ' Разместить два дочерних узла под дочерним узлом
    ' узла верхнего уровня
    For lCount = 1 To 2
        With oDiagramNodeChild.Children.AddNode
            .Shape.TextFrame.Characters.Font.Name = _
                wksDiagrams.Range(sRANGE_LEVELS).Cells(3, 2).Text
            .Shape.Fill.ForeColor.SchemeColor = _
                wksDiagrams.Range(sRANGE_LEVELS).Cells(3, 3).Value
        End With
    Next lCount

    ' Создать еще один дочерний узел ниже узла верхнего уровня
    Set oDiagramNodeChild = .Children.AddNode

    With oDiagramNodeChild
        .Shape.TextFrame.Characters.Font.Name = _
            wksDiagrams.Range(sRANGE_LEVELS).Cells(2, 2).Text
        .Shape.Fill.ForeColor.SchemeColor = _
            wksDiagrams.Range(sRANGE_LEVELS).Cells(2, 3).Value
    End With

    ' Добавить два дочерних узла ниже этого дочернего узла
    '(он находится на один уровень ниже узла верхнего уровня)
    For lCount = 1 To 2
```

```
        With oDiagramNodeChild.Children.AddNode
            .Shape.TextFrame.Characters.Font.Name = _
                wksDiagrams.Range(sRANGE_LEVELS).Cells(3, 2).Text
            .Shape.Fill.ForeColor.SchemeColor = _
                wksDiagrams.Range(sRANGE_LEVELS).Cells(3, 3).Value
        End With
    Next lCount

End With

End Sub
```

Объект DiagramNode и коллекция DiagramNodes

Объект DiagramNode представляет одну геометрическую фигуру внутри объекта Diagram. Геометрические узлы ниже определенного узла являются дочерними. Для добавления узлов ниже текущего узла воспользуйтесь методом AddNode свойства Children этого объекта.

Коллекция DiagramNodes содержит все объекты Node из объекта Diagram. Каждый объект Node соответствует геометрической фигуре внутри объекта Diagram.

Общие свойства коллекции DiagramNodes

Определения свойств Application, Creator и Parent приводились в начале приложения.

Свойства коллекции DiagramNodes

Имя	*Тип возвращаемого значения*	*Описание*
Count	Long	Только чтение. Возвращает количество объектов внутри коллекции

Методы коллекции DiagramNodes

Имя	*Тип возвращаемого значения*	*Параметры*	*Описание*
Item	Diagram Node	Index As Variant	Возвращает один объект из коллекции
SelectAll			Выделяет все геометрические фигуры в коллекции

Общие свойства объекта DiagramNode

Определения свойств Application, Creator и Parent приводились в начале приложения.

Свойства объекта DiagramNode

Имя	*Тип возвращаемого значения*	*Описание*
Children	DiagramNode-Children	Только чтение. Возвращает коллекцию дочерних узлов для конкретного узла
Diagram	IMsoDiagram	Только чтение. Возвращает представление диаграммы

Имя	*Тип возвращаемого значения*	*Описание*
Layout	MsoOrgChart-LayoutType	Устанавливает/возвращает стиль форматирования дочерних узлов организационной диаграммы
Root	DiagramNode	Только чтение. Возвращает корень корневого узла диаграммы
Shape	Shape	Только чтение. Возвращает геометрическую фигуру, связанную с указанным комментарием, узлом диаграммы или гиперссылкой
TextShape	Shape	Только чтение. Возвращает геометрическую фигуру текстового поля, связанного с узлом диаграммы

Методы объекта DiagramNode

Имя	*Тип возвращаемого значения*	*Параметры*	*Описание*
AddNode	DiagramNode	[pos As MsoRelativeNodePosition], [nodeType As MsoDiagramNodeType]	Создает узел диаграммы и возвращает объект DiagramNode, представляющий созданный узел
CloneNode	DiagramNode	copyChildren As Boolean, [pTargetNode As DiagramNode], [pos As MsoRelativeNode-Position]	Клонирует узел диаграммы и возвращает объект DiagramNode, представляющий клонированный узел
Delete			Удаляет объект
MoveNode		pTargetNode As DiagramNode, pos As MsoRelativeNodePosition	Перемещает узел диаграммы и все его дочерние узлы в пределах диаграммы
NextNode	DiagramNode		Выделяет следующий узел диаграммы в последовательности узлов и возвращает объект DiagramNode, представляющий выделенный узел
PrevNode	DiagramNode		Возвращает предыдущий узел диаграммы в коллекции узлов диаграммы
ReplaceNode		pTargetNode As DiagramNode	Заменяет целевой узел диаграммы исходным
SwapNode		pTargetNode As DiagramNode, [swapChildren As Boolean]	Меняет местами целевой и исходный узлы диаграммы
Transfer-Children		pReceiving Node As DiagramNode	Дочерние узлы исходной диаграммы перемещаются в получающий узел

Объект DiagramNodeChildren

Объект DiagramNodeChildren представляет родительскую геометрическую форму, расположенную на один уровень ниже объекта DiagramNode. Каждый объект DiagramNodeChildren сам по себе является объектом DiagramNode. Если объект DiagramNodeChildren имеет дочерние узлы ниже по иерархии, каждый из этих узлов также может рассматриваться, как объект DiagramNodeChildren.

Общие свойства объекта DiagramNodeChildren

Определения свойств Application, Creator и Parent приводились в начале приложения.

Свойства объекта DiagramNodeChildren

Имя	*Тип возвращаемого значения*	*Описание*
Count	Long	Только чтение. Возвращает количество объектов в коллекции
FirstChild	DiagramNode	Только чтение. Возвращает первый дочерний узел родительского узла
LastChild	DiagramNode	Только чтение. Возвращает последний дочерний узел родительского узла

Методы объекта DiagramNodeChildren

Имя	*Тип возвращаемого значения*	*Параметры*	*Описание*
AddNode	DiagramNode	[Index], [nodeType As MsoDiagramNodeType]	Создает новый объект DiagramNode
SelectAll			Выделяет все геометрические фигуры в пределах коллекции

Объект Dialog и коллекция Dialogs

Коллекция Dialogs предоставляет доступ к списку всех встроенных диалоговых окон Excel. Константы XlBuiltinDialog используются для получения доступа к отдельным объектам Dialog в пределах коллекции Dialogs. Объект Dialog представляет одно встроенное диалоговое окно Excel. Каждый объект Dialog обладает собственными свойствами в зависимости от типа диалогового окна. Кроме типичных атрибутов, коллекция Dialogs предоставляет свойство Count, возвращающее количество объектов Dialog в коллекции.

Общие свойства объекта Dialog

Определения свойств Application, Creator и Parent приводились в начале приложения.

Методы объекта Dialog

Имя	Тип возвращаемого значения	Параметры	Описание
Show	Boolean	[Arg1], [Arg2],... [Arg30]	Отображает диалоговое окно с использованием указанных параметров. Если пользователь щелкает на кнопке ОК, метод возвращает значение True. Если пользователь щелкает на кнопке Отмена (Cancel), метод возвращает значение False. Список передаваемых аргументов зависит от типа диалогового окна

Пример: объект Dialog и коллекция Dialogs

```
Sub ShowPrinterSelection()
  ' Показать диалоговое окно выбора принтера
  Application.Dialogs(xlDialogPrinterSetup).Show
End Sub
```

Объект DisplayUnitLabel

Объект DisplayUnitLabel содержит весь текст и форматирование, связанные с метками единиц измерения на осях диаграмм. Например, если значения на осях измеряются в миллионах, отображение таких больших чисел может привести к переполнению экранного пространства ненужной информацией. Применение такой метки единицы измерения, как "Миллион", позволяет использовать меньшие числа. Для этого объекта родительским является объект Axis. Этот объект обычно используется вместе со свойством HasDisplayUnit родительского объекта Axis.

Общие свойства объекта DisplayUnitLabel

Определения свойств Application, Creator и Parent приводились в начале приложения.

Свойства объекта DisplayUnitLabel

Имя	Тип возвращаемого значения	Описание
AutoScaleFont	Variant	Устанавливает/возвращает необходимость автоматического изменения размера шрифта при изменении размера родительской диаграммы
Border	Border	Только чтение. Возвращает свойства границы вокруг метки единицы измерения
Caption	String	Устанавливает/возвращает текст метки единицы измерения
Characters	Characters	Только чтение. Параметры: [Start], [Length]. Возвращает объект, содержащий все символы метки единицы измерения. Этот объект позволяет управлять отдельными символами метки
Fill	ChartFillFormat	Только чтение. Возвращает объект, содержащий параметры форматирования заливки для метки единицы измерения
Font	Font	Только чтение. Возвращает объект, содержащий параметры шрифта для метки единицы измерения

Имя	*Тип возвращаемого значения*	*Описание*
Horizontal-Alignment	Variant	Устанавливает/возвращает параметры горизонтального выравнивания метки единицы измерения. Для установки этого свойства используются константы xlAlign
Interior	Interior	Только чтение. Возвращает объект, содержащий параметры форматирования области текста метки единицы измерения (например, внутреннего цвета)
Left	Double	Устанавливает/возвращает расстояние от левого края текстовой области метки единицы измерения до левого края диаграммы
Name	String	Только чтение. Возвращает имя объекта DisplayUnit-Label
Orientation	Variant	Устанавливает/возвращает угол отображения текста в метке единицы измерения. Значение свойства может указываться в градусах (в диапазоне от -90 до 90) или с помощью одной из констант XlOrientation
ReadingOrder	Long	Устанавливает/возвращает направление чтения текста (слева направо или справа налево). Это свойство может использоваться с некоторыми языками пользовательского интерфейса
Shadow	Boolean	Устанавливает/возвращает необходимость создания эффекта тени для метки единицы измерения
Text	String	Устанавливает/возвращает текст метки единицы измерения
Top	Double	Устанавливает/возвращает расстояние от верхнего края области текста метки единицы измерения до верхнего края диаграммы
Vertical-Alignment	Variant	Устанавливает/возвращает параметры вертикального выравнивания метки единицы измерения. Для установки этого свойства можно воспользоваться константами xlVAlign

Методы объекта DisplayUnitLabel

Имя	*Тип возвращаемого значения*	*Параметры*	*Описание*
Delete	Variant		Удаляет метку единицы измерения
Select	Variant		Выделяет метку единицы измерения на диаграмме

Пример: объект DisplayUnitLabel

```
Sub AddUnitLabel()
   Dim oDUL As DisplayUnitLabel
   ' Отформатировать ось Y для отображения метки единицы измерения
   With Charts(1).Axes(xlValue)
      .DisplayUnit = xlThousands
      .HasDisplayUnitLabel = True
      ' Получить метку единицы измерения
     Set oDUL = .DisplayUnitLabel
   End With
   ' Отформатировать метку единицы измерения
With oDUL
```

```
        .Caption = "Тысячи"
        .Font.Name = "Arial"
        .VerticalAlignment = xlCenter
    End With
End Sub
```

Объект DownBars

Объект DownBars содержит параметры полос понижения на диаграмме. Для объекта DownBars родительским является объект Chart. Для проверки существования этого объекта можно воспользоваться свойством HasUpDownBars объекта ChartGroup.

Общие свойства объекта DownBars

Определения свойств Application, Creator и Parent приводились в начале приложения.

Свойства объекта DownBars

Имя	*Тип возвращаемого значения*	*Описание*
Border	Border	Только чтение. Возвращает свойства границы вокруг полос понижения
Fill	ChartFillFormat	Только чтение. Возвращает объект, содержащий параметры форматирования заливки для полос понижения
Interior	Interior	Только чтение. Возвращает объект, содержащий параметры форматирования внутренней области полос понижения (например, цвета внутренней области)
Name	String	Только чтение. Возвращает имя полос понижения

Методы объекта DownBars

Имя	*Тип возвращаемого значения*	*Параметры*	*Описание*
Delete	Variant		Удаляет полосы понижения
Select	Variant		Выделяет полосы понижения на диаграмме

Объект DropLines

Этот объект содержит информацию о форматировании линий проекции на диаграмме. Для него родительским является объект ChartGroup. Для проверки существования этого объекта можно воспользоваться свойством HasDropLines объекта ChartGroup.

Общие свойства объекта DropLines

Определения свойств Application, Creator и Parent приводились в начале приложения.

Свойства объекта DropLines

Имя	Тип возвращаемого значения	Описание
Border	Border	Только чтение. Возвращает свойства границы вокруг линий проекции
Name	String	Только чтение. Возвращает имя линий проекции

Методы объекта DropLines

Имя	Тип возвращаемого значения	Параметры	Описание
Delete	Variant		Удаляет линии проекции
Select	Variant		Выделяет линии проекции на диаграмме

Пример: объект DropLines

```
Sub AddAndFormatDropLines()
   Dim oDLine As DropLines
   ' Показать drop lines
   Charts(1).ChartGroups(1).HasDropLines = True
   ' Получить объект DropLines
   Set oDLine = Charts(1).ChartGroups(1).DropLines
   ' Отформатировать линии проекции
   With oDLine
       .Border.Weight = xlMedium
       .Border.LineStyle = xlDash
       .Border.ColorIndex = 3
   End With
End Sub
```

Объект Error и коллекция Errors

В объекте Error хранится одна ошибка из ячейки, содержащей несколько потенциальных ошибок.

В коллекции Errors хранятся все ошибки из такой ячейки. Каждая ячейка может содержать несколько ошибок.

Эти ошибки аналогичны возможности проверки ошибок из Excel 2003. Различные типы определяемых в Excel ошибок можно обнаружить на вкладке Проверка ошибок (Error Checking) в диалоговом окне Сервис⇨Параметры (Tools⇨Параметры). В приложениях Excel содержащие ошибки ячейки выделяются с помощью небольшого треугольника в верхнем левом углу. По умолчанию на большинстве систем используется зеленый цвет индикатора, но цвет индикатора можно изменить с помощью параметра Цвет отображения ошибок (Error Indicator Color) на вкладке Проверка ошибок (Error Checking) диалогового окна Параметры (Options).

При выделении диапазона, содержащего ошибку, отображается диаграмма в виде знака восклицания в желтом ромбе. После этого пользователь может щелкнуть на пиктограмме и выбрать способ обработки ошибки в этом диапазоне. Если был выбран способ обработки, например, игнорирование или один из предложенных вариантов, зеленый индикатор исчезает во всех диапазонах, содержащих подобную ошибку. Зеленый индикатор в других ячейках указывает на то, что в них присутствуют ошибки других типов.

На момент написания этой книги объект коллекции Errors и объект Error не поддерживали обработку нескольких ошибок в диапазоне из нескольких ячеек (как описано

выше). В справочном руководстве и в руководстве по объектной модели показано, что родительским объектом для коллекции Errors является объект Range. Но каждая попытка обратиться к коллекции Errors из объекта Range, содержащего несколько ячеек, приводит к сообщению об ошибке. Так как любая ячейка может содержать несколько ошибок, в коллекции Errors хранятся все ошибки, содержащиеся в пределах одной ячейки. В результате для обработки ошибок в диапазоне из нескольких ячеек придется перебирать ячейки в цикле.

Обратите внимание, что ни коллекция Errors, ни объект Error не содержат счетчика или бинарного значения, указывающих на факт существования хотя бы одной ячейки. По этой причине потребуется дополнительный код, который будет перебирать все типы ошибок в каждой интересующей ячейке с проверкой значения Value объекта Error. Если это свойство возвращает значение True, ошибка соответствующего типа присутствует в ячейке.

Для перебора типов ошибок при определении их наличия можно воспользоваться свойством Item коллекции Errors.

Общие свойства коллекции Errors

Определения свойств Application, Creator и Parent приводились в начале приложения.

Свойства коллекции Errors

Имя	*Тип возвращаемого значения*	*Описание*
Item	Error	Возвращает объект Error, хранящийся в коллекции Errors

Общие свойства объекта Error

Определения свойств Application, Creator и Parent приводились в начале приложения.

Свойства объекта Error

Имя	*Тип возвращаемого значения*	*Описание*
Ignore	Boolean	Устанавливает/возвращает необходимость обработки ошибок в пределах диапазона
Value	Boolean	Только чтение. Подтверждает соответствие всем критериям проверки

Объект ErrorBars

В объекте ErrorBars хранятся параметры форматирования полос ошибки для диаграммы. Для этого объекта родительским является объект SeriesCollection.

Общие свойства объекта ErrorBars

Определения свойств Application, Creator и Parent приводились в начале приложения.

Свойства объекта ErrorBars

Имя	*Тип возвращаемого значения*	*Описание*
Border	Border	Только чтение. Возвращает свойства границ вокруг полос ошибки
EndStyle	XlEndStyleCap	Устанавливает/возвращает стиль, который используется для окончаний полос ошибки
Name	String	Только чтение. Возвращает имя полос ошибки

Методы объекта ErrorBars

Имя	*Тип возвращаемого значения*	*Параметры*	*Описание*
ClearFormats	Variant		Сбрасывает форматирование полос ошибки
Delete	Variant		Удаляет полосы ошибки
Select	Variant		Выделяет полосы ошибки на диаграмме

Пример: объект ErrorBars

```
Sub AddAndFormatErrorBars()
   Dim oSer As Series
   Dim oErrBars As ErrorBars
   ' Добавляет полосы ошибки к первой последовательности
   ' (на уровне +/- 10% от значения)
   Set oSer = Charts(1).SeriesCollection(1)
   oSer.ErrorBar xlY, xlErrorBarIncludeBoth, _
              xlErrorBarTypePercent, 10
   ' Получить объект ErrorBars
   Set oErrBars = oSer.ErrorBars
   ' Установить форматирование полос ошибки
   With oErrBars
      .Border.Weight = xlThick
      .Border.LineStyle = xlContinuous
      .Border.ColorIndex = 7
      .EndStyle = xlCap
   End With
End Sub
```

Коллекция ErrorCheckingOptions

Представляет все параметры, доступные на вкладке Проверка ошибок (Error Checking) в диалоговом окне Сервис⇨Параметры (Tools⇨Options). Свойство BackgroupChecking этого объекта позволяет скрыть все индикаторы ошибок (небольшой треугольник в верхнем правом углу ячейки).

Свойства этого объекта позволяют сообщить Excel тип необходимых проверок на наличие ошибок.

На объект ErrorCheckingOptions можно ссылаться через объект Application, а значит изменение его свойств затронет все открытые окна.

Общие свойства коллекции ErrorCheckingOptions

Определения свойств Application, Creator и Parent приводились в начале приложения.

Свойства коллекции ErrorCheckingOptions

Имя	*Тип возвращаемого значения*	*Описание*
`BackgroundChecking`	`Boolean`	Устанавливает/возвращает необходимость фоновой проверки наличия ошибок. Определяет, отображается ли кнопка автоматического исправления возле содержащей ошибку ячейки
`EmptyCellReferences`	`Boolean`	Устанавливает/возвращает необходимость проверки наличия ошибок для ячеек, содержащих формулу, которая ссылается на пустую ячейку
`EvaluateToError`	`Boolean`	Устанавливает/возвращает необходимость проверки наличия ошибок для ячеек, значение которых оценивается, как ошибочное
`InconsistentFormula`	`Boolean`	Устанавливает/возвращает необходимость проверки наличия ошибок в ячейках, содержащих противоречивую формулу
`IndicatorColorIndex`	`XlColorIndex`	Устанавливает/возвращает цвет индикатора ошибки
`ListDataValidation`	`Boolean`	Возвращает значение `True`, если в списке включена проверка действительности данных
`NumberAsText`	`Boolean`	Устанавливает/возвращает необходимость проверки наличия ошибок для чисел, записанных в текстовой форме
`OmittedCells`	`Boolean`	Устанавливает/возвращает необходимость проверки наличия ошибок в ячейках, содержащих формулы, которые ссылаются на диапазон с пропущенными соседними ячейками (которые можно включить в диапазон)
`TextDate`	`Boolean`	Устанавливает/возвращает необходимость проверки наличия ошибок в ячейках, которые содержат текстовое представление даты с двумя цифрами для записи года
`UnlockedFormulaCells`	`Boolean`	Устанавливает/возвращает необходимость проверки наличия ошибок для разблокированных ячеек, содержащих формулу

Пример: коллекция ErrorCheckingOptions

В следующей подпрограмме для установки параметров проверки наличия ошибок используется таблица на листе.

```
Sub SetErrorCheckingOptions()

    Dim rngSettings As Range
    Dim vSetting As Variant

    ' Найти начало таблицы Settings
    Set rngSettings = wksErrors.Range("ErrorSettings")

    ' Перебрать все свойства ErrorChecking и
    ' установить их в соответствиями со значениями в таблице.
    With Application.ErrorCheckingOptions

        vSetting = rngSettings.Cells(1, 2).Value
        If Len(vSetting) And (vSetting = True Or vSetting = False) Then
            .BackgroundChecking = vSetting
        End If
```

```
        vSetting = rngSettings.Cells(2, 2).Value
        If Len(vSetting) And (vSetting = True Or vSetting = False) Then
            .EvaluateToError = vSetting
        End If

        vSetting = rngSettings.Cells(3, 2).Value
        If Len(vSetting) And (vSetting = True Or vSetting = False) Then
            .TextDate = vSetting
        End If

        vSetting = rngSettings.Cells(4, 2).Value
        If Len(vSetting) And (vSetting = True Or vSetting = False) Then
            .NumberAsText = vSetting
        End If

        vSetting = rngSettings.Cells(5, 2).Value
        If Len(vSetting) And (vSetting = True Or vSetting = False) Then
            .InconsistentFormula = vSetting
        End If

        vSetting = rngSettings.Cells(6, 2).Value
        If Len(vSetting) And (vSetting = True Or vSetting = False) Then
            .OmittedCells = vSetting
        End If

        vSetting = rngSettings.Cells(7, 2).Value
        If Len(vSetting) And (vSetting = True Or vSetting = False) Then
            .UnlockedFormulaCells = vSetting
        End If

        vSetting = rngSettings.Cells(8, 2).Value
        If Len(vSetting) And (vSetting = True Or vSetting = False) Then
            .EmptyCellReferences = vSetting
        End If

        vSetting = rngSettings.Cells(9, 2).Value
        If LCase(vSetting) = "xlcolorindexautomatic" Then
            .IndicatorColorIndex = xlColorIndexAutomatic
        ElseIf Len(vSetting) And (vSetting > 1 And vSetting < 100) Then
            .IndicatorColorIndex = vSetting
        End If

    End With

    ' Иногда индикаторы не отображаются
    ' после завершения подпрограммы.
    ' В таком случае необходимо обновить содержимое экрана.
    Application.ScreenUpdating = True
End Sub
```

Объект FillFormat

Объект FillFormat описывает эффекты заливки, доступные для геометрических фигур. Например, объект FillFormat определяет сплошную, текстурированную и мозаичную заливку родительской геометрической фигуры. Получить доступ к объекту FillFormat можно только через объект Shape.

Общие свойства объекта FillFormat

Определения свойств Application, Creator и Parent приводились в начале приложения.

Свойства FillFormat

Имя	*Тип возвращаемого значения*	*Описание*
BackColor	ColorFormat	Только чтение. Возвращает цвет фона в виде объекта ColorFormat
ForeColor	ColorFormat	Только чтение. Возвращает основной цвет в виде объекта ColorFormat
Gradient-ColorType	MsoGradient-ColorType	Только чтение. Возвращает используемый тип градиентной заливки цветом
Gradient-Degree	Single	Только чтение. Возвращает яркость градиентной заливки
Gradient-Style	MsoGradient-Style	Только чтение. Возвращает ориентацию используемого градиента
Gradient-Variant	Integer	Только чтение. Возвращает используемый вариант градиентной заливки из центра
Pattern	MsoPattern-Type	Только чтение. Возвращает шаблон заливки, если используется этот тип заливки
Preset-GradientType	MsoPreset-GradientType	Только чтение. Возвращает тип применяемой градиентной заливки
Preset-Texture	MsoPreset-Texture	Только чтение. Возвращает встроенную текстуру, которая используется для заливки
TextureName	String	Только чтение. Возвращает имя собственной текстуры, которая используется для заливки
TextureType	MsoTextureType	Только чтение. Возвращает тип используемой текстуры: собственная, встроенная или смешанная
Transparency	Single	Устанавливает/возвращает степень прозрачности заливки. Принимает значения в диапазоне от 0 (непрозрачная) до 1 (полностью прозрачная)
Type	MsoFillType	Только чтение. Возвращает тип используемой заливки: текстура, градиент, сплошной цвет, фон (в Excel не применяется), изображение или смешанный
Visible	MsoTriState	Устанавливает/возвращает видимость параметров заливки в родительской геометрической фигуре

Методы объекта FillFormat

Имя	*Тип возвращаемого значения*	*Параметры*	*Описание*
OneColor-Gradient		Style As MsoGradientStyle, Variant As Integer, Degree as Single	Устанавливает стиль, вариант и степень градиентной заливки одним цветом
Patterned		Pattern As MsoPatternType	Устанавливает шаблон заливки

Имя	*Тип возвращаемого значения*	*Параметры*	*Описание*
`PresetGradient`		`Style As MsoGradientStyle, Variant As Integer, PresetGradientType As MsoPresetGradientType`	Выбирает стиль, вариант и встроенный тип градиента, используемые для градиентной заливки
`PresetTextured`		`PresetTexture As MsoPresetTexture`	Устанавливает встроенную текстуру для заливки
`Solid`			Устанавливает цвет для сплошной заливки
`TwoColorGradient`		`Style As MsoGradientStyle, Variant As Integer`	Устанавливает стиль двухцветной градиентной заливки
`UserPicture`		`PictureFile As String`	Выбирает изображение для заливки из файла `PictureFile`
`UserTextured`		`TextureFile As String`	Устанавливает собственную текстуру для заливки. Текстура выбирается из файла `TextureFile`

Пример: объект FillFormat

```
Sub FormatShape()
   Dim oFF As FillFormat
   ' Получить форматирование заливки первой геометрической фигуры
   Set oFF = ActiveSheet.Shapes(1).Fill
   ' Отформатировать геометрическую фигуру
   With oFF
       .TwoColorGradient msoGradientFromCorner, 1
       .ForeColor.SchemeColor = 3
       .BackColor.SchemeColor = 5
   End With
End Sub
```

Объект Filter и коллекция Filters

В коллекции `Filters` хранятся все фильтры, связанные с родительским объектом `AutoFilter`. Каждый объект `Filter` определяет один фильтр для одного столбца диапазона с установленным автофильтром. Родительским объектом этой коллекции является объект `AutoFilter`.

Кроме типичных атрибутов коллекции, коллекция `Filters` предоставляет еще одно свойство. Свойство `Count` возвращает количество объектов `Filter`, которые хранятся в коллекции.

Общие свойства объекта Filter

Определения свойств `Application`, `Creator` и `Parent` приводились в начале приложения.

Свойства объекта Filter

Имя	*Тип возвращаемого значения*	*Описание*
`Criteria1`	`Variant`	Только чтение. Возвращает первый определенный для фильтра критерий (например, ">=5")
`Criteria2`	`Variant`	Только чтение. Возвращает второй критерий для фильтра, если такой критерий определен
`On`	`Boolean`	Только чтение. Возвращает информацию об использовании фильтра в данный момент
`Operator`	`XlAutoFilter-Operator`	Только чтение. Возвращает тип оператора сортировки, определенного для фильтра (например, `xlTop10Items`)

Пример: объект Filter и коллекция Filters

Пример использования объекта `Filter` и коллекции `Filters` приводился в разделе, посвященном объекту `AutoFormat`.

Объект Floor

Объект `Floor` содержит параметры форматирования базового уровня трехмерной диаграммы. Родительским для этого объекта является объект `Chart`.

Общие свойства объекта Floor

Определения свойств `Application`, `Creator` и `Parent` приводились в начале приложения.

Свойства объекта Floor

Имя	*Тип возвращаемого значения*	*Описание*
`Border`	`Border`	Только чтение. Возвращает свойства границы вокруг базового уровня трехмерной диаграммы
`Fill`	`ChartFill-Format`	Только чтение. Возвращает объект, содержащий параметры форматирования заливки базового уровня трехмерной диаграммы
`Interior`	`Interior`	Только чтение. Возвращает объект, содержащий параметры форматирования внутренней области базового уровня диаграммы (например, цвет внутренней области)
`Name`	`String`	Только чтение. Возвращает имя объекта `Floor`
`PictureType`	`Variant`	Устанавливает/возвращает способ отображения связанного рисунка на базовом уровне трехмерной диаграммы (например, с растягиванием или в виде мозаики). Для установки этого свойства используются константы `XlPictureType`

Методы объекта Floor

Имя	*Тип возвращаемого значения*	*Параметры*	*Описание*
`ClearFormats`	`Variant`		Сбрасывает форматирование объекта `Floor`

Имя	*Тип возвращаемого значения*	*Параметры*	*Описание*
Paste			Вставляет рисунок из буфера обмена в объекта Floor
Select	Variant		Выделяет базовый уровень на диаграмме

Пример: объект Floor

```
Sub FormatFloor()
   Dim oFlr As Floor
   ' Получить объект Floor для диаграммы
   Set oFlr = Charts(1).Floor
   ' Отформатировать базовый уровень с использованием текстуры
   ' белого мрамора.
   With oFlr
      .Fill.PresetTextured msoTextureWhiteMarble
      .Fill.Visible = True
   End With
End Sub
```

Объект Font

В объекте Font хранятся все атрибуты форматирования, связанные со шрифтами в родительском объекте, включая тип, размер и цвет шрифта. Возможными родительскими объектами объекта Font могут выступать объекты AxisTitle, Characters, ChartArea, ChartTitle, DataLabel, Legend, LegendEntry, Range, Style и TickLabel. Коллекция DataLabels также может выступать в роли родительского объекта для объекта Font.

Общие свойства объекта Font

Определения свойств Application, Creator и Parent приводились в начале приложения.

Свойства объекта Font

Имя	*Тип возвращаемого значения*	*Описание*
Background	Variant	Устанавливает/возвращает тип фона, который используется для текста (xlBackgroundAutomatic, xlBackgroundOpaque и xlBackgroundTransparent). Для установки этого свойства применяются константы XlBackground. Используется только для текста на диаграммах
Bold	Variant	Устанавливает/возвращает необходимость использования полужирного шрифта
Color	Variant	Устанавливает/возвращает цвет шрифта. Для создания значения цвета необходимо использовать функцию RGB
ColorIndex	Variant	Устанавливает/возвращает цвет шрифта. Для создания значения цвета используются константы XlColorIndex или значение индекса в текущей цветовой палитре
FontStyle	Variant	Устанавливает/возвращает стиль шрифта (например, “Полужирный”)

Имя	*Тип возвращаемого значения*	*Описание*
Italic	Variant	Устанавливает/возвращает необходимость использования курсива
Name	Variant	Устанавливает/возвращает имя шрифта
OutlineFont	Variant	Устанавливает/возвращает необходимость использования контурного шрифта. Не применяется в операционной системе Windows
Shadow	Variant	Устанавливает/возвращает необходимость использования шрифта с эффектом тени. Не применяется в операционной системе Windows
Size	Variant	Устанавливает/возвращает размер шрифта
Strike-through	Variant	Устанавливает/возвращает необходимость использования перечеркнутого шрифта
Subscript	Variant	Устанавливает/возвращает необходимость использования шрифта в виде нижнего индекса
Superscript	Variant	Устанавливает/возвращает необходимость применения шрифта в виде верхнего индекса
Underline	Variant	Устанавливает/возвращает необходимость применения подчеркнутого шрифта

Пример: объект Font

```
Sub FormatCellFont()
   Dim oFont As Font
   ' Получить шрифт выделенного диапазона
   Set oFont = Selection.Font
   ' Указать формат шрифта
   With oFont
      .Name = "Times New Roman"
      .Size = 16            ' Пункты
      .ColorIndex = 5       ' Синий
      .Bold = True
      .Underline = xlSingle
   End With
End Sub
```

Объект FormatCondition и коллекция FormatConditions

В коллекции FormatConditions хранится условное форматирование определенного диапазона ячеек. Родительским объектом в данном случае выступает объект Range. В коллекции FormatConditions может храниться до трех объектов FormatCondition. Каждый объект FormatCondition представляет определенное форматирование, которое применяется при выполнении условия.

Кроме стандартных атрибутов, коллекция FormatConditions предоставляет одно свойство и два метода. Свойство Count возвращает количество объектов FormatCondition, которые хранятся в коллекции. Метод Add может использоваться для добавления в коллекцию условия форматирования. При этом необходимо указать значение параметра Type (для установки этого параметра используются константы XlFormatConditionType). Условие определяется с помощью параметров Operator, Formula1 и Formula2.

Имя	*Тип возвращаемого значения*	*Описание*
`Count`	`Long`	Только чтение. Возвращает количество объектов, хранящихся в коллекции
`Add`	`Format-Condition`	Метод. Параметры: `Type As XlFormatConditionType`, `[Operator]`, `[Formula1]`, `[Formula2]`. Добавляет объект `FormatCondition` в коллекцию
`Delete`		Метод. Удаляет объект `FormatCondition` из коллекции

Общие свойства объекта FormatCondition

Определения свойств `Application`, `Creator` и `Parent` приводились в начале приложения.

Свойства объекта FormatCondition

Имя	*Тип возвращаемого значения*	*Описание*
`Borders`	`Borders`	Только чтение. Возвращает коллекцию с отдельными атрибутами границ для условия форматирования
`Font`	`Font`	Только чтение. Возвращает объект, содержащий параметры шрифта для условия форматирования
`Formula1`	`String`	Только чтение. Возвращает значение ячейки, выражение или формулу, которые должны быть равны `True/False`. Если формула или выражение равны значению `True`, форматирование применяется к ячейке
`Formula2`	`String`	Только чтение. Возвращает значение ячейки, выражение или формулу, которые должны быть равны значениям `True/False`. Это свойство действительно только в случае, если параметр `Operator` установлен в значение `xlBetween` или `xlNotBetween`
`Interior`	`Interior`	Только чтение. Возвращает объект, содержащий параметры форматирования внутренней области условия форматирования (например, цвета внутренней области)
`Operator`	`Long`	Только чтение. Возвращает оператор, который применяется к свойствам `Formula1` и `Formula2`. Значения этого свойства совпадают со значениями констант `XlFormat-ConditionOperator`
`Type`	`Long`	Только чтение. Возвращает необходимость применения форматирования из объекта `FormatCondition` в зависимости от значения ячейки или в зависимости от формулы. Значения этого свойства совпадают со значениями констант `XlFormatConditionType`

Методы объекта FormatCondition

Имя	*Тип возвращаемого значения*	*Параметры*	*Описание*
Delete			Удаляет условие форматирования
Modify		Type As XlFormatConditionType, [Operator], [Formula1], [Formula2]	Единственный способ модификации условия форматирования, так как все свойства объекта предназначены только для чтения

Пример: объект FormatCondition и коллекция FormatConditions

```
Sub AddConditionalFormat()
   Dim oFC As FormatCondition
   ' Удалить существующие условия
   For Each oFC In Selection.FormatConditions
      Selection.FormatConditions(1).Delete
   Next
   ' Добавить первое условие
   Set oFC = Selection.FormatConditions.Add(Type:=xlCellValue, _
Operator:=xlLess, Formula1:="10")
   With oFC
       .Font.ColorIndex = 2                  ' белый
       .Font.Bold = True
       .Interior.Pattern = xlSolid
       .Interior.Color = RGB(255, 0, 0)      ' красный
   End With
   ' Добавить второе условие
   Set oFC = Selection.FormatConditions.Add(Type:=xlCellValue, _
Operator:=xlBetween, Formula1:="10", Formula2:="40")
   With oFC
       .Font.Color = RGB(0, 255, 0)
       .Font.Bold = False
       .Interior.Pattern = xlNone
   End With
   ' Добавить третье условие
   Set oFC = Selection.FormatConditions.Add(Type:=xlCellValue, _
Operator:=xlGreater, Formula1:="40")
With oFC
       .Font.Color = RGB(0, 0, 255)
       .Font.Bold = True
       .Interior.Pattern = xlNone
   End With
End Sub
```

Объект FreeformBuilder

Объект FreeformBuilder используется родительским объектом Shape для создания новых геометрических фигур произвольной формы. Для получения объекта FreeformBuilder применяется метод BuildFreeform объекта Shape.

Общие свойства FreeformBuilder

Определения свойств Application, Creator и Parent приводились в начале приложения.

Методы объекта FreeformBuilder

Имя	Тип возвращаемого значения	Параметры	Описание
AddNodes		SegmentType As MsoSegmentType, EditingType As MsoEditingType, X1 As Single, Y1 As Single, [X2], [Y2], [X3], [Y3]	Добавляет точку к геометрической фигуре, которая рисуется в данный момент Добавляемый узел связывается линией с последним добавленным узлом. Параметр SegmentType применяется для описания типа линии между узлами. Параметры X1, Y1, X2, Y2, X3, Y3 используются для определения положения добавляемого узла. Координаты отсчитываются от верхнего левого угла документа
ConvertToShape	Shape		Выполняет преобразование добавленных узлов в объект Shape

Пример: объект FreeformBuilder

```
Sub MakeArch()
   Dim oFFB As FreeformBuilder
   ' Создать новый объект FreeformBuilder
   Set oFFB = ActiveSheet.Shapes.BuildFreeform(msoEditingCorner, 100, 300)
   ' Добавить линии в произвольную фигуру
   With oFFB
      .AddNodes msoSegmentLine, msoEditingAuto, 100, 200
      .AddNodes msoSegmentCurve, msoEditingCorner, _
150, 150, 0, 0, 200, 200
      .AddNodes msoSegmentLine, msoEditingAuto, 200, 300
      .AddNodes msoSegmentLine, msoEditingAuto, 100, 300
      ' Преобразовать узлы в геометрическую фигуру
      .ConvertToShape
   End With
End Sub
```

Объект Graphic

Этот объект представляет изображение, которое можно разместить в одной из шести точек верхнего или нижнего колонтитула страницы. Этот объект одновременно является аналогом кнопок Вставить рисунок и Форматировать рисунок в диалоговых окнах Верхний колонтитул (Header) и Нижний колонтитул (Footer) в диалоговом окне Параметры страницы (Page Setup).

Важно обратить внимание, что изменение свойств этого объекта не приведет к добавлению объекта в область верхнего или нижнего колонтитула страницы. Для вывода рисунка в одну из областей колонтитулов необходимо добавить строку “&G” (с помощью кода VBA).

Общие свойства объекта Graphic

Определения свойств Application, Creator и Parent приводились в начале приложения.

Свойства объекта Graphic

Имя	*Тип возвращаемого значения*	*Описание*
`Brightness`	`Single`	Устанавливает/возвращает яркость указанного рисунка. Значение этого свойства должно находиться в диапазоне от 0.0 (минимальная яркость) до 1.0 (максимальная яркость)
`ColorType`	`MsoPictureColorType`	Устанавливает/возвращает преобразование цвета, которое применяется к указанному рисунку или объекту OLE
`Contrast`	`Single`	Устанавливает/возвращает контраст указанного рисунка. Значение этого свойства должно находиться в диапазоне от 0.0 (минимальный контраст) до 1.0 (максимальный контраст)
`CropBottom`	`Single`	Устанавливает/возвращает количество точек, которые вырезаются из нижней части указанного рисунка или объекта OLE
`CropLeft`	`Single`	Устанавливает/возвращает количество точек, которые вырезаются из левой части указанного рисунка или объекта OLE
`CropRight`	`Single`	Устанавливает/возвращает количество точек, которые вырезаются из правой части указанного рисунка или объекта OLE
`CropTop`	`Single`	Устанавливает/возвращает количество точек, которые вырезаются из верхней части рисунка или объекта OLE
`Filename`	`String`	Устанавливает/возвращает адрес URL или путь к указанному объекту
`Height`	`Single`	Устанавливает/возвращает высоту объекта
`LockAspectRatio`	`MsoTriState`	Устанавливает/возвращает необходимость сохранения пропорций при изменении размера геометрической фигуры
`Width`	`Single`	Устанавливает/возвращает ширину объекта

Пример: объект Graphic

Следующая подпрограмма выдает запрос на выбор графического файла. Выбранный рисунок помещается в верхний колонтитул активного листа в виде водяного знака. Размер рисунка меняется в соответствии с размерами страницы

```
Sub AddWatermark()
    Dim oSheet As Object
    Dim sFile As String

    On Error Resume Next
        Set oSheet = ActiveSheet
    On Error GoTo 0

    ' Убедиться в существовании активного листа
    If Not oSheet Is Nothing Then

        ' Установить свойства диалогового окна Открыть файл
        With Application.FileDialog(msoFileDialogFilePicker)

            ' Изменить принятый по умолчанию заголовок
            ' диалогового окна
            .Title = "Вставить рисунок в центр колонтитула"
```

```
            ' Разрешить выбор только одного файла
            .AllowMultiSelect = False

            ' Очистить фильтры и создать собственный фильтр
            ' Переключиться на собственный фильтр перед
            ' отображением диалогового окна
            .Filters.Add "All Pictures", "*.gif; *.jpg; *.jpeg; _
             *.bmp; *.wmf; *.gif;*.emf;*.dib;*.jfif;*.jpe",1

            ' Разрешить предварительный просмотр рисунков
            .InitialView = msoFileDialogViewThumbnail

            ' Показать диалоговое окно
            '-1 означает, что пользователь не щелкал на кнопке Отмена
            If .Show = -1 Then
                ' Сохранить имя выбранного файла
                sFile = .SelectedItems(1)

                ' Вставить рисунок в верхний колонтитул
                With oSheet.PageSetup
                    With .CenterHeaderPicture
                        .Filename = sFile
                        .ColorType = msoPictureWatermark
                        .LockAspectRatio = True

                        ' Рисунок должен заполнять всю страницу
                        ' Предполагается размер листа Letter
                        ' Ориентация альбомная
                        .Width = Application.InchesToPoints(17)
                    End With

                    'Отобразить графику
                    'Без этого ничего не произойдет
                    .CenterHeader = "&G"
                End With

            End If

           'Удалить фильтр после завершения
           .Filters.Clear

        End With


    End If
End Sub
```

Объект Gridlines

В этом объекте хранятся параметры форматирования, связанные с основными и дополнительными направляющими линиями осей диаграммы. Направляющие линии являются расширением отметок на оси, видимым на фоне диаграммы. Они упрощают пользователям определение значения объекта диаграммы. Родительским для объекта Gridlines является объект Axis. Для проверки действительности объекта и создания объекта Gridlines воспользуйтесь свойствами HasMajorGridlines и HasMinorGridlines объекта Axis.

Общие свойства объекта Gridlines

Определения свойств Application, Creator и Parent приводились в начале приложения.

Свойства объекта Gridlines

Имя	*Тип возвращаемого значения*	*Описание*
Border	Border	Только чтение. Возвращает свойства границы вокруг направляющих линий
Name	String	Только чтение. Возвращает имя объекта Gridlines

Методы объекта Gridlines

Имя	*Тип возвращаемого значения*	*Параметры*	*Описание*
Delete	Variant		Удаляет объект Gridlines
Select	Variant		Выделяет направляющие линии на диаграмме

Пример: объект Gridlines

```
Sub FormatGridlines()
   Dim oGL As Gridlines
   ' Удостовериться, что ось Y имеет направляющие линии
   With Charts(1).Axes(xlValue)
       .HasMajorGridlines = True

       ' Получить объект Gridlines для основных направляющих линий
       Set oGL = .MajorGridlines
   End With
' Отформатировать направляющие линии
   With oGL
       .Border.ColorIndex = 5
       .Border.LineStyle = xlDash
       .Border.Weight = xlThin
   End With
End Sub
```

Коллекция GroupShapes

В коллекции GroupShapes хранятся все геометрические фигуры, составляющие сгруппированную геометрическую фигуру. Коллекция GroupShapes содержит коллекцию объектов Shape. Родительским объектом для этой коллекции является объект Shape.

Кроме типичных атрибутов коллекции, коллекция GroupShapes предоставляет два свойства. Свойство Count возвращает количество объектов Shape внутри коллекции, а свойство Range возвращает подмножество геометрических фигур из коллекции Shapes.

Объект HiLoLines

В объекте HiLoLines хранятся параметры форматирования для линий максимума-минимума диаграммы. Родительским для объекта HiLoLines является объект ChartGroup. Линии максимума-минимума соединяют точки наибольших и наименьших значений в группе двумерной линейной диаграммы.

Общие свойства объекта HiLoLines

Определения свойств Application, Creator и Parent приводились в начале приложения.

Свойства объекта HiLoLines

Имя	*Тип возвращаемого значения*	*Описание*
Border	Border	Только чтение. Возвращает свойства границы вокруг линий максимума-минимума
Name	String	Только чтение. Возвращает имя объекта HiLoLines

Методы HiLoLines

Имя	*Тип возвращаемого значения*	*Параметры*	*Описание*
Delete	Variant		Удаляет линии максимума-минимума
Select	Variant		Выделяет линии максимума-минимума на диаграмме

Пример: объект HiLoLines

```
Sub AddAndFormatHiLoLines()
   Dim oHLL As HiLoLines
   ' Добавить линии максимума-минимума к первой группе
Charts(1).ChartGroups(1).HasHiLoLines = True
   ' Получить объект HiLoLines
   Set oHLL = Charts(1).ChartGroups(1).HiLoLines
   ' Отформатировать линии
   With oHLL
      .Border.Weight = xlMedium
      .Border.LineStyle = xlContinuous
      .Border.ColorIndex = 3
   End With
End Sub
```

Объект HPageBreak и коллекция HPageBreaks

В коллекции HPageBreak хранятся горизонтальные разрывы страницы из области печати родительского объекта. Каждый объект HPageBreak представляет один горизонтальный разрыв страницы в области печати родительского объекта. В качестве родительских для объекта HPageBreaks могут выступать объекты Worksheet и Chart.

Кроме типичных атрибутов, коллекция HPageBreaks предоставляет одно свойство и один метод. Свойство Count возвращает количество объектов HPageBreak в пределах коллекции. Метод Add позволяет добавить в коллекцию объект HPageBreak (при этом на лист добавляется горизонтальный разрыв страницы). Метод Add принимает параметр Before, описывающий диапазон, над которым вставляется горизонтальный разрыв страницы.

Общие свойства объекта HPageBreak

Определения свойств Application, Creator и Parent приводились в начале приложения.

Свойства объекта HPageBreak

Имя	Тип возвращаемого значения	Описание
Extent	XlPageBreak-Extent	Только чтение. Возвращает необходимость растягивания горизонтального разрыва страницы на весь экран или только в пределах области печати
Location	Range	Устанавливает/возвращает ячейку, в которой расположен горизонтальный разрыв страницы. Верхний край ячейки совпадает с линией разрыва страницы
Type	XlPageBreak	Устанавливает/возвращает тип разрыва страницы (автоматический или ручной)

Методы объекта HPageBreak

Имя	Тип возвращаемого значения	Параметры	Описание
Delete			Удаляет разрыв страницы
DragOff		Direction As XlDirection, RegionIndex As Long	Вытаскивает разрыв страницы за пределы области печати. Параметр Direction описывает направление перетаскивания разрыва страницы. Параметр RegionIndex определяет область печати, в которую перетаскивается разрыв страницы

Пример: объект HPageBreak и коллекция HPageBreaks

```
Sub AddHPageBreaks()
   Dim oCell As Range
   ' Перебрать все ячейки первого столбца
   For Each oCell In ActiveSheet.UsedRange.Columns(1).Cells
      ' Если размер шрифта в ячейке равен 16, добавить разрыв страницы
      ' над ячейкой
      If oCell.Font.Size = 16 Then
         ActiveSheet.HPageBreaks.Add oCell
      End If
   Next
End Sub
```

Объект Hyperlink и коллекция Hyperlinks

Коллекция Hyperlinks предоставляет доступ к списку гиперссылок на листе или в диапазоне. Каждый объект Hyperlink соответствует одной гиперссылке на листе или в диапазоне. Кроме типичных атрибутов, коллекция Hyperlinks предоставляет методы Add и Delete. В качестве значения параметра Anchor метод Add принимает текст или изображение, которые будут использоваться в качестве ссылки, а в качестве параметра Address метод принимает адрес URL или имя файла. На основе этой информации создается объект Hyperlink. Метод Delete удаляет объект Hyperlink из коллекции. Кроме этого, коллекция Hyperlinks предоставляет свойство Count, которое возвращает количество объектов Hyperlink в коллекции.

Общие свойства объекта Hyperlink

Определения свойств Application, Creator и Parent приводились в начале приложения.

Свойства объекта Hyperlink

Имя	*Тип возвращаемого значения*	*Описание*
Address	String	Устанавливает/возвращает имя файла или адрес URL, на которые ссылается гиперссылка
EmailSubject	String	Устанавливает/возвращает тему сообщения электронной почты, если гиперссылка указывает на адрес электронной почты
Name	String	Только чтение. Возвращает имя гиперссылки
Range	Range	Только чтение. Возвращает положение гиперссылки в документе
ScreenTip	String	Устанавливает/возвращает текст, который отображается при наведении указателя мыши на гиперссылку
Shape	Shape	Только чтение. Возвращает геометрическую фигуру, связанную с гиперссылкой
SubAddress	String	Устанавливает/возвращает точку в документе, связанную с гиперссылкой
TextToDisplay	String	Устанавливает/возвращает текст, который отображается в виде гиперссылки
Type	Long	Устанавливает/возвращает тип объекта

Методы объекта Hyperlink

Имя	*Тип возвращаемого значения*	*Параметры*	*Описание*
AddToFavorites			Добавляет значение свойства Address в папку Избранное (Favorites)
CreateNewDocument		Filename As String, EditNow As Boolean, Overwrite As Boolean	Создает новый документ. Имя файла (FileName) извлекается из адреса гиперссылки. Для открытия документа в соответствующем редакторе устанавливается свойство EditNow в значение True. Для перезаписи существующих документов с таким же именем устанавливается свойство Overwrite в значение True
Delete			Удаляет объект Hyperlink
Follow		[NewWindow], [AddHistory], [ExtraInfo], [Method], [HeaderInfo]	Открывает документ, на который указывает свойство Address. Установка параметра NewWindow в значение True позволяет открыть документ в новом окне. Для отображения документа в папке журнала необходимо установить параметр AddHistory в значение True. Параметр Method позволяет установить свойство ExtraInfo в значение Get или Post

Пример: объект Hyperlink и коллекция Hyperlinks

В этом примере на базе гиперссылок создается лист "Содержание".

```
Sub CreateHyperlinkTOC()
   Dim oBk As Workbook
   Dim oShtTOC As Worksheet, oSht As Worksheet
   Dim iRow As Integer
   Set oBk = ActiveWorkbook
   ' Добавить в книгу новый лист
   Set oShtTOC = oBk.Worksheets.Add
   With oShtTOC
      ' Добавить в лист заголовок
      .Range("A1").Value = "Содержание"
      ' Добавить гиперссылки на адреса
      ' электронной почты и web-страницы
      .Hyperlinks.Add .Range("A3"), "mailto:Me@MyISP.com", _
                    TextToDisplay:="Отправить комментарии по электронной
↳почте"
      .Hyperlinks.Add .Range("A4"), "http://www.wrox.com", _
                    TextToDisplay:="Посетить сайт Wrox Press"
   End With
   ' Перебрать все листы в книге
   ' добавляя гиперссылки
   iRow = 6
   For Each oSht In oBk.Worksheets
   If oSht.Name <> oShtTOC.Name Then
         oShtTOC.Hyperlinks.Add oShtTOC.Cells(iRow, 1), "", _
                    SubAddress:="'" & oSht.Name & "'!A1", _
                    TextToDisplay:=oSht.Name
         iRow = iRow + 1
      End If
   Next
End Sub
```

Объект Interior

В объекте `Interior` хранятся параметры форматирования внутренней области родительского объекта. В качестве родительских объектов могут выступать объекты `AxisTitle`, `ChartArea`, `ChartObject`, `ChartTitle`, `DataLabel`, `DownBars`, `Floor`, `FormatCondition`, `Legend`, `LegendKey`, `OLEObject`, `PlotArea`, `Point`, `Range`, `Series`, `Style`, `Upbars` и `Walls`. Кроме этого, в качестве родительских объектов могут выступать коллекции `ChartObjects`, `DataLabels` и `OLEObjects`.

Общие свойства объекта Interior

Определения свойств `Application`, `Creator` и `Parent` приводились в начале приложения.

Свойства объекта Interior

Имя	*Тип возвращаемого значения*	*Описание*
`Color`	`Variant`	Устанавливает/возвращает цвет внутренней области. В данном случае для создания значения цвета используется функция `RGB`
`ColorIndex`	`Variant`	Устанавливает/возвращает цвет внутренней области. В данном случае используются константы `XlColorIndex` или номер цвета в текущей цветовой палитре

Имя	*Тип возвращаемого значения*	*Описание*
`InvertIfNegative`	`Variant`	Устанавливает/возвращает необходимость инвертирования цвета внутренней области родительского объекта, если значение цвета является отрицательным
`Pattern`	`Variant`	Устанавливает/возвращает шаблон, который используется для заполнения внутренней области родительского объекта. Для установки этого свойства используются константы `XlPattern`
`PatternColor`	`Variant`	Устанавливает/возвращает цвет шаблона внутренней области. В данном случае для создания значения цвета используется функция `RGB`
`PatternColorIndex`	`Variant`	Устанавливает/возвращает цвет шаблона внутренней области. В данном случае используются константы `XlColorIndex` или номер цвета в текущей цветовой палитре

Пример: объект Interior

```
Sub FormatRange()
   Dim oInt As Interior
   ' Получить внутреннюю область текущего выделения
   Set oInt = Selection.Interior
' Отформатировать внутреннюю область для использования сплошного
' желтого цвета
   '(цвет зависит от текущей цветовой палитры книги)
   With oInt
      .Pattern = xlSolid
      .ColorIndex = 6
   End With
End Sub
```

Объект IRtdServer

Этот объект предоставляет возможность подключения к серверу данных реального времени (Real-Time Data Server – RTD). Такие серверы позволяют Excel регулярно получать данные без необходимости создания дополнительного кода. В предыдущих версиях Excel регулярное обновление данных требовало использования метода `OnTime`. Сервер данных реального времени отправляет обновленную информацию автоматически. Интервал времени устанавливается на сервере или с помощью метода `HeartbeatInterval` объекта `IRTDUpdateEvent`.

Этот объект по своей сути напоминает функцию листа RTD, которая регулярно отображает данные в ячейке листа.

Обратите внимание, что перед использованием необходимо создать экземпляр этого объекта с помощью ключевого слова Implement.

Методы объекта IRtdServer

Имя	*Тип возвращаемого значения*	*Параметры*	*Описание*
`ConnectData`		`TopicID, Strings, GetNewValues`	Метод вызывается при открытии файла, который содержит функции данных реального времени, или при вводе формулы, содержащей функцию `RTD`

Имя	Тип возвращаемого значения	Параметры	Описание
DisconnectData		TopicID	Применяется для уведомления сервера данных реального времени о том, что тема больше не используется
Heartbeat	Long		Проверяет активность сервера данных реального времени. Отрицательное значение или ноль указывают на неудачную попытку подключения к серверу. Положительное значение подтверждает успешность попытки подключения
RefreshData		ByRef TopicCount As Long	Вызывается для получения новых данных после уведомления от сервера данных реального времени о наличии обновлений
ServerStart	Long	CallbackObject	Вызывается сразу после создания экземпляра сервера данных реального времени. Отрицательное значение или ноль указывают на неудачную попытку подключения к серверу. Положительное значение подтверждает успешность попытки подключения
ServerTerminate			Используется для завершения подключения к серверу

Объект IRTDUpdateEvent

Этот объект представляет события обновления в реальном времени. Объект `IRTDUpdateEvent` используется для установки интервала между обновлениями для объекта `IRtdServer`. Интервал устанавливается с помощью свойства `HeartbeatInterval`. Объект `IRTDUpdateEvent` возвращается в результате использования метода `ServerStart` объекта `IRtdServer` для подключения к серверу данных реального времени.

Свойства объекта IRTDUpdateEvent

Имя	Тип возвращаемого значения	Описание
HeartbeatInterval	Long	Устанавливает/возвращает интервал между обновлениями данных реального времени

Методы объекта IRTDUpdateEvent

Имя	Тип возвращаемого значения	Параметры	Описание
Disconnect			Заставляет сервер данных реального времени отключиться от указанного объекта
UpdateNotify			Приложение Excel получает уведомление от сервера данных реального времени о получении новых данных

Объект LeaderLines

В объекте LeaderLines хранятся атрибуты форматирования, связанные с линиями меток данных на диаграмме. Эти линии связывают метки данных с маркерами данных. Родительским для объекта LeaderLines является объект Series. Для создания объекта LeaderLines и проверки его существования используется свойство HasLeaderLines объекта Series.

Общие свойства LeaderLines

Определения свойств Application, Creator и Parent приводились в начале приложения.

Свойства объекта LeaderLines

Имя	*Тип возвращаемого значения*	*Описание*
Border	Border	Только чтение. Возвращает свойства границы вокруг линий меток

Методы объекта LeaderLines

Имя	*Тип возвращаемого значения*	*Параметры*	*Описание*
Delete			Удаляет объект LeaderLines
Select			Выделяет метки данных на диаграмме

Пример: объект LeaderLines

```
Sub AddAndFormatLeaderLines()
   Dim oLL As LeaderLines
   ' Использование первой последовательности круговой диаграммы
   With Charts(1).SeriesCollection(1)
      ' Добавить метки линиями мекто (при необходимости)
      .ApplyDataLabels HasLeaderLines:=True

      ' Расположить метки
      .DataLabels.Position = xlLabelPositionBestFit

      ' Получить объект LeaderLines. Если все метки расположены
      ' на принятых по умолчанию позициях, будет выдано сообщение
      ' об ошибке
      Set oLL = .LeaderLines
   End With
   ' Отформатировать линии меток
   With oLL
      .Border.LineStyle = xlContinuous
      .Border.ColorIndex = 5
   End With
End Sub
```

Объект Legend

В объекте Legend хранятся параметры форматирования и записи легенды для определенной диаграммы. Родительским для объекта Legend является объект Chart. Для создания или проверки существования объекта Legend можно воспользоваться свойством HasLegend объекта Chart.

Общие свойства объекта Legend

Определения свойств Application, Creator и Parent приводились в начале приложения.

Свойства объекта Legend

Имя	*Тип возвращаемого значения*	*Описание*
AutoScale-Font	Variant	Устанавливает/возвращает необходимость автоматического изменения размера шрифта при изменении размера родительской диаграммы
Border	Border	Только чтение. Возвращает свойства границы вокруг легенды
Fill	ChartFill-Format	Только чтение. Возвращает объект, содержащий параметры форматирования заливки легенды диаграммы
Font	Font	Только чтение. Возвращает объект, содержащий параметры шрифта для текста легенды
Height	Double	Устанавливает/возвращает высоту поля легенды
Interior	Interior	Только чтение. Возвращает объект, содержащий параметры форматирования внутренней области легенды (например, цвет внутренней области)
Left	Double	Устанавливает/возвращает расстояние от левого края поля легенды до левого края родительской диаграммы
Name	String	Только чтение. Возвращает имя объекта Legend
Position	XlLegend-Position	Устанавливает/возвращает положение легенды на диаграмме (например, xlLegendPositionCorner, xlLegendPositionLeft)
Shadow	Boolean	Устанавливает/возвращает необходимость отображения эффекта тени для легенды
Top	Double	Устанавливает/возвращает расстояние от верхнего края поля легенды до верхнего края родительской диаграммы
Width	Double	Устанавливает/возвращает ширину поля легенды

Методы объекта Legend

Имя	*Тип возвращаемого значения*	*Параметры*	*Описание*
Clear	Variant		Очищает легенду
Delete	Variant		Удаляет легенду
Legend-Entries	Object	[Index]	Возвращает один объект LegendEntry или коллекцию LegendEntries в зависимости от наличия параметра Index. Содержит весь текст и маркеры легенды
Select	Variant		Выделяет легенду на диаграмме

Пример: объект Legend

```
Sub PlaceLegend()
   Dim oLgnd As Legend
   ' Удостовериться, что диаграмма содержит легенду
```

```
    Charts(1).HasLegend = True
    ' Получить объект Legend
    Set oLgnd = Charts(1).Legend
    ' Расположить и отформатировать легенду
    With oLgnd
        .Position = xlLegendPositionRight
        .Border.LineStyle = xlNone
        .AutoScaleFont = False
    End With
End Sub
```

Объект LegendEntry и коллекция LegendEntries

В коллекции LegendEntries хранятся записи легенды. Каждый объект LegendEntry представляет одну запись легенды. Запись состоит из текста и маркера. Текст всегда связан с именем последовательности или именем линии тренда. Родительским для коллекции LegendEntries является объект Legend.

Кроме типичных атрибутов, коллекция LegendEntries предоставляет одно свойство. Свойство Count возвращает количество объектов LegendEntry, которые хранятся в коллекции.

Общие свойства объекта LegendEntry

Определения свойств Application, Creator и Parent приводились в начале приложения.

Свойства объекта LegendEntry

Имя	*Тип возвращаемого значения*	*Описание*
AutoScaleFont	Variant	Устанавливает/возвращает необходимость автоматического изменения размера шрифта при изменении размера родительской диаграммы
Font	Font	Только чтение. Возвращает объект, содержащий параметры шрифта для текста записи легенды
Height	Double	Только чтение. Возвращает высоту записи легенды
Index	Long	Только чтение. Возвращает положение объекта LegendEntry в коллекции LegendEntries
Left	Double	Только чтение. Возвращает расстояние от левого края записи легенды до левого края диаграммы
LegendKey	LegendKey	Только чтение. Возвращает объект, содержащий форматирование маркера записи легенды
Top	Double	Только чтение. Возвращает расстояние от верхнего края записи легенды до верхнего края родительской диаграммы
Width	Double	Только чтение. Возвращает ширину записи легенды

Методы объекта LegendEntry

Имя	*Тип возвращаемого значения*	*Параметры*	*Описание*
Delete	Variant		Удаляет объект LegendEntry
Select	Variant		Выделяет запись легенды на диаграмме

Пример: объект LegendEntry и коллекция LegendEntries

```
Sub FormatLegendEntries()
   Dim oLE As LegendEntry
   ' Добавить легенду на диаграмму
   Charts(1).HasLegend = True
   ' Перебрать все записи легенды
   For Each oLE In Charts(1).Legend.LegendEntries
      ' Отформатировать каждую запись с использованием
      ' собственного стиля шрифта
      With oLE
         .Font.Size = 10 + .Index * 4
         .Font.Bold = (.Index Mod 2) = 0
         .Font.ColorIndex = .Index
      End With
   Next
End Sub
```

Объект LegendKey

В объекте `LegendKey` хранятся свойства и методы, позволяющие управлять форматированием ключевого маркера записи легенды. Ключевой маркер является визуальным представлением, например цветом, который идентифицирует определенную последовательность или линию тренда.

Общие свойства объекта LegendKey

Определения свойств `Application`, `Creator` и `Parent` приводились в начале приложения.

Свойства объекта LegendKey

Имя	*Тип возвращаемого значения*	*Описание*
`Border`	`Border`	Только чтение. Возвращает свойства границы вокруг ключа легенды
`Fill`	`ChartFillFormat`	Только чтение. Возвращает объект, содержащий параметры форматирования заливки для ключа легенды последовательности или линии тренда
`Height`	`Double`	Только чтение. Возвращает высоту ключа легенды
`Interior`	`Interior`	Только чтение. Возвращает объект, содержащий параметры форматирования внутренней области ключа легенды (например, цвета внутренней области)
`InvertIfNegative`	`Boolean`	Устанавливает/возвращает необходимость инвертирования цвета ключа легенды, если значение цвета является отрицательным
`Left`	`Double`	Только чтение. Возвращает расстояние от левого края ключа легенды до левого края родительской диаграммы
`MarkerBackgroundColor`	`Long`	Устанавливает/возвращает цвет фона ключа легенды. В данном случае для создания значения цвета должна использоваться функция `RGB`
`MarkerBackgroundColorIndex`	`XlColorIndex`	Устанавливает/возвращает цвет фона ключа легенды. В данном случае для установки значения свойства могут использоваться константы `XlColorIndex` или значение индекса цвета в текущей цветовой палитре

Имя	*Тип возвращаемого значения*	*Описание*
MarkerForegroundColor	Long	Устанавливает/возвращает основной цвет ключа легенды. В данном случае для создания значения цвета должна использоваться функция RGB
MarkerForegroundColorIndex	XlColorIndex	Устанавливает/возвращает основной цвет ключа легенды. В данном случае для установки значения свойства могут использоваться константы XlColorIndex или значение индекса цвета в текущей цветовой палитре
MarkerSize	Long	Устанавливает/возвращает размер маркера ключа легенды
MarkerStyle	XlMarkerStyle	Устанавливает/возвращает тип маркера, который будет использоваться в качестве ключа легенды (например, квадрат, ромб, треугольник, рисунок и т.д.)
PictureType	Long	Устанавливает/возвращает способ отображения связанного рисунка в легенде (например, со сжатием, в виде мозаики). Для установки значения этого свойства можно воспользоваться константами XlPictureType
PictureUnit	Long	Устанавливает/возвращает количество единиц, которые представлены рисунком, если свойство PictureType установлено в значение xlScale
Shadow	Boolean	Устанавливает/возвращает необходимость отображения эффекта тени вокруг ключа записи легенды
Smooth	Boolean	Устанавливает/возвращает необходимость сглаживания кривых при отображении ключа легенды
Top	Double	Только чтение. Возвращает расстояние от верхнего края ключа записи легенды до верхнего края родительской диаграммы
Width	Double	Только чтение. Возвращает ширину ключа записи легенды

Методы объекта LegendKey

Имя	*Тип возвращаемого значения*	*Параметры*	*Описание*
ClearFormats	Variant		Очищает форматирование, установленное с помощью объекта LegendKey
Delete	Variant		Удаляет объект LegendKey
Select	Variant		Выделяет ключ легенды на родительской диаграмме

Пример: объект LegendKey

```
Sub FormatLegendKeys()
   Dim oLE As LegendEntry
   Dim oLK As LegendKey
   ' Добавить легенду в диаграмму
   Charts(1).HasLegend = True
   ' Перебрать все записи легенды
   For Each oLE In Charts(1).Legend.LegendEntries
      ' Получить ключ легенды для записи
      Set oLK = oLE.LegendKey
```

```
    ' Отформатировать каждый ключ легенды своим цветом и размером
    With oLK
        .MarkerForegroundColor = oLE.Index
        .MarkerSize = oLE.Index 2 + 1
    End With
  Next
End Sub
```

Объект LineFormat

Объект LineFormat описывает форматирование линии родительского объекта Shape. Для получения доступа к объекту LineFormat применяется свойство Line объекта Shape. Обычно объект LineFormat используется для изменения таких свойств линии, как стили конечной стрелки и направление стрелки.

Общие свойства объекта LineFormat

Определения свойств Application, Creator и Parent приводились в начале приложения.

Свойства объекта LineFormat

Имя	*Тип возвращаемого значения*	*Описание*
BackColor	ColorFormat	Только чтение. Возвращает объект, позволяющий управлять цветом фона линии
BeginArrowheadLength	MsoArrowheadLength	Устанавливает/возвращает длину наконечника стрелки в начале линии
BeginArrowheadStyle	MsoArrowheadStyle	Устанавливает/возвращает внешний вид наконечника стрелки в начале линии
BeginArrowheadWidth	MsoArrowheadWidth	Устанавливает/возвращает ширину наконечника стрелки в начале линии
DashStyle	MsoLineDashStyle	Устанавливает/возвращает стиль линии
EndArrowheadLength	MsoArrowheadLength	Устанавливает/возвращает длину наконечника стрелки в конце линии
EndArrowheadStyle	MsoArrowheadStyle	Устанавливает/возвращает внешний вид наконечника стрелки в конце линии
EndArrowheadWidth	MsoArrowheadWidth	Устанавливает/возвращает ширину наконечника стрелки в конце линии
ForeColor	ColorFormat	Только чтение. Возвращает объект, поддерживающий управление цветом фона линии
Pattern	MsoPatternType	Устанавливает/возвращает шаблон, который используется для рисования линии
Style	MsoLineStyle	Устанавливает/возвращает стиль линии
Transparency	Single	Устанавливает/возвращает прозрачность (1) или непрозрачность (0) линии
Visible	MsoTriState	Устанавливает/возвращает возвращает видимость линии
Weight	Single	Устанавливает/возвращает толщину линии

Пример: объект LineFormat

```
Sub AddAndFormatLine()
   Dim oShp As Shape
   Dim oLF As LineFormat
   ' Добавить линию
   Set oShp = ActiveSheet.Shapes.AddLine(100, 100, 200, 250)
   ' Получить объекта формата линии
   Set oLF = oShp.Line
   ' Установить формат линии
   With oLF
      .BeginArrowheadStyle = msoArrowheadOval
      .EndArrowheadStyle = msoArrowheadTriangle
      .EndArrowheadLength = msoArrowheadLong
      .EndArrowheadWidth = msoArrowheadWide
      .Style = msoLineSingle
   End With
End Sub
```

Объект LinkFormat

Объект `LinkFormat` описывает атрибуты связывания для объекта `OLE` или рисунка. Объект `LinkFormat` связан с объектом `Shape`. Только объекты `Shape`, которые являются действительными объектами OLE, могут получать доступ к объекту `LineFormat`.

Общие свойства объекта LinkFormat

Определения свойств `Application`, `Creator` и `Parent` приводились в начале приложения.

Свойства объекта LinkFormat

Имя	*Тип возвращаемого значения*	*Описание*
`AutoUpdate`	`Boolean`	Устанавливает/возвращает необходимость обновления родительского объекта Shape при каждом изменении файла источника или при открытии родительского объекта
`Locked`	`Boolean`	Устанавливает/возвращает отказ от обновления объекта `Shape` из исходного файла

Методы LinkFormat

Имя	*Тип возвращаемого значения*	*Параметры*	*Описание*
`Update`			Обновляет родительский объект `Shape` из исходных данных

Пример: объект LinkFormat

```
Sub UpdateShapeLinks()
   Dim oShp As Shape
   Dim oLnkForm As LinkFormat
   ' Перебрать все геометрические фигуры на листе
   For Each oShp In ActiveSheet.Shapes
      ' Является ли фигура объектом OLE?
      If oShp.Type = msoLinkedOLEObject Or oShp.Type = msoLinkedPicture
Then
```

```
            ' Если да, указать формат связи
            Set oLnkForm = oShp.LinkFormat
            ' и обновить связь.
            oLnkForm.Update
        End If
    Next
End Sub
```

Объект ListColumn

Этот объект описывает столбец в списке. В коллекции ListColumns хранятся все столбцы списка, каждый из которых имеет собственный объектListColumn.

Общие свойства объекта ListColumn

Определения свойств Application, Creator и Parent приводились в начале приложения.

Свойства объекта ListColumn

Имя	*Тип возвращаемого значения*	*Описание*
Index	Integer	Номер объекта ListColumn в списке
ListDataFormat	ListDataFormatObject	Возвращает объект ListDataFormat для объекта ListColumn
Name	String	Возвращает имя объекта
Range		Возвращает диапазон, к которому относится объект ListColumn из текущего списка
SharePointFormula	String	Возвращает строку, которая используется в качестве формулы вычислимого поля
TotalsCalculation		Определяет тип расчетов в строке итогов
XPath		Возвращает объект XPath для элемента, связанного с определенным диапазоном

Методы объекта ListColumn

Имя	*Тип возвращаемого значения*	*Параметры*	*Описание*
Delete			Удаляет столбец данных из списка

Объект ListDataFormat

В объекте ListDataFormat хранятся все свойства типа данных для объекта ListColumn.

Общие свойства ListDataFormat

Определения свойств Application, Creator и Parent приводились в начале приложения.

Свойства объекта ListDataFormat

Имя	*Тип возвращаемого значения*	*Описание*
AllowFillIn	Boolean	Определяет возможность ввода значений пользователем или выбор определенных значений из списка
Choices	StringArray	Содержит строковый массив возможных вариантов
Decimal-Places	Long	Определяет количество отображаемых десятичных знаков
DefaultValue	Variant	Принятое по умолчанию значение
IsPercent	Boolean	Определяет необходимость отображения этого свойства в виде процентного значения
LCID	Long	Определяет используемый символ валюты
Max-Characters	Long	Максимальное количество вводимых символов
MaxNumber	Variant	Наибольшее используемое число
MinNumber	Variant	Наименьшее используемое число
ReadOnly	Boolean	Определяет возможность изменения значения элемента
Required	Boolean	Определяет, является ли элемент обязательным
Type	XlListDataType	Используется при связывании элемента с сайтом SharePoint

Объект ListObject

Этот объект соответствует списку в пределах листа.

Свойства объекта ListObject

Имя	*Тип возвращаемого значения*	*Описание*
Application	Object	Возвращает объект Application, представляющий приложение Excel
Count	Integer	Возвращает количество элементов в коллекции
Creator	Long	Возвращает значение типа Long, которое описывает создавшее объект приложение. Это значение может быть преобразовано в шестнадцатеричную строку и в строку ASCII
Item	Object	Возвращает один объект из коллекции объектов
Parent	Object	Возвращает ссылку на родительский объект

Методы объекта ListObject

Имя	*Тип возвращаемого значения*	*Параметры*	*Описание*
Delete			Удаляет элемент из коллекции
Publish	String	[Target - StringArray] [LinkSource - Boolean]	Публикует объект на сервере Windows Sharepoint. Строковый массив Target должен содержать следующие элементы: 0 — адрес URL сервера SharePoint; 1 — имя списка; 2 — описание списка

Имя	*Тип возвращаемого значения*	*Параметры*	*Описание*
Refresh			Обновляет текущие данные с помощью сервера Windows Sharepoint
Resize		Range	Позволяет изменить размер объекта ListObject до размера определенного диапазона
Unlink			Удаляет текущую связь с сервером Windows Sharepoint
UpdateChange			Обновляет всю информацию с сервера Windows Sharepoint с помощью синхронизации всех изменений

Объект ListRow

Как подсказывает имя, этот объект представляет строку в списке.

Свойства объекта ListRow

Имя	*Тип возвращаемого значения*	*Описание*
Application	Object	Возвращает объект Application, представляющий приложение Excel
Creator	Long	Возвращает значение типа Long, которое описывает создавшее объект приложение. Это значение может быть преобразовано в шестнадцатеричную строку и в строку ASCII
Index	Long	Возвращает номер объекта в коллекции
InvalidData	Boolean	Возвращает значение типа Boolean, которое определяет, прошла ли строка проверку на действительность значения
Parent	Object	Возвращает ссылку на родительский объект
Range	Range	Возвращает диапазон, к которому относится текущий объект

Методы объекта ListRow

Имя	*Тип возвращаемого значения*	*Параметры*	*Описание*
Delete			Удаляет элемент из коллекции

Объект Mailer

Объект Mailer используется на компьютерах Macintosh для отправки файлов Excel средствами PowerTalk Mailer.

Общие свойства объекта Mailer

Определения свойств Application, Creator и Parent приводились в начале приложения.

Свойства объекта Mailer

Имя	*Тип возвращаемого значения*	*Описание*
BCCRecipients	Variant	Устанавливает/возвращает список скрытых копий
CCRecipients	Variant	Устанавливает/возвращает список копий
Enclosures	Variant	Устанавливает/возвращает список вложений
Received	Boolean	Только чтение. Указывает, было ли получено сообщение
SendDateTime	Date	Только чтение. Возвращает дату и время отправки сообщения
Sender	String	Только чтение. Возвращает имя отправителя сообщения электронной почты
Subject	String	Устанавливает/возвращает строку темы сообщения электронной почты
ToRecipients	Variant	Устанавливает/возвращает массив имен получателей
WhichAddress	Variant	Устанавливает/возвращает адрес, с которого было отправлено сообщение электронной почты

Объект Name и коллекция Names

В коллекции Names хранится список именованных диапазонов в пределах книги. Каждый объект Name описывает диапазон ячеек в пределах книги, доступ к которому возможен по имени. Некоторые объекты Name являются встроенными (например, Print_Area). Остальные объекты определяются пользователем. В качестве родительского объекта коллекции Names может выступать объект Workbook, Application и Worksheet. Кроме этого, доступ к объекту Name можно получить через объект Range.

Кроме типичных атрибутов, коллекция Names предоставляет метод Add. Этот метод добавляет в коллекцию объект Name. Параметры метода Add соответствуют свойствам объекта Name.

Общие свойства объекта Name

Определения свойств Application, Creator и Parent приводились в начале приложения.

Свойства объекта Name

Имя	*Тип возвращаемого значения*	*Описание*
Category	String	Устанавливает/возвращает категорию объекта Name на языке, который использовался для создания макроса. Это свойство действительно только для объектов Name, которые являются определенной пользователем функцией или командой
Category-Local	String	Устанавливает/возвращает категорию объекта Name на языке конечного пользователя. Это свойство действительно только для объектов Name, которые являются определенной пользователем функцией или командой

Имя	*Тип возвращаемого значения*	*Описание*
Index	Long	Только чтение. Возвращает положение объекта Name в коллекции Names
MacroType	XlXLMMacroType	Устанавливает/возвращает, ссылается ли объект Name на команду, функцию или просто на диапазон
Name	String	Устанавливает/возвращает имя объекта Name на языке макроса
NameLocal	String	Устанавливает/возвращает имя объекта Name на языке конечного пользователя
RefersTo	Variant	Устанавливает/возвращает текстовую ссылку на диапазон, на который ссылается объект Name, на языке макроса в формате A1
RefersTo-Local	Variant	Устанавливает/возвращает текстовую ссылку на диапазон, на который ссылается объект Name, на языке пользователя в формате A1
RefersToR1C1	Variant	Устанавливает/возвращает текстовую ссылку на диапазон, на который ссылается объект Name, на языке макроса в формате R1C1
RefersTo-R1C1Local		Устанавливает/возвращает текстовую ссылку на диапазон, на который ссылается объект Name, на языке пользователя в формате R1C1
RefersTo-Range	Range	Только чтение. Возвращает диапазон, на который ссылается объект Range
ShortcutKey	String	Устанавливает/возвращает комбинацию клавиш, которая запускает макрос Microsoft Excel 4.0, связанный с объектом Name
Value	String	Устанавливает/возвращает текстовую ссылку на диапазон, на который ссылается объект Name, на языке макроса в формате A1
Visible	Boolean	Устанавливает/возвращает необходимость отображения имени объекта Name в диалоговом окне Имена (Names)

Методы объекта Name

Имя	*Тип возвращаемого значения*	*Параметры*	*Описание*
Delete			Удаляет объект Name из коллекции

Пример: объект Name и коллекция Names

```
Sub DeleteInvalidNames()
   Dim oName As Name
   ' Перебрать все имена в активной книге
   For Each oName In ActiveWorkbook.Names
      ' Является ли имя действительным?
      If InStr(1, oName.RefersTo, "#REF") > 0 Then

         ' Если нет, выдать сообщение об ошибке
         Debug.Print "Удалено имя " & oName.Name & " - " & _
oName.RefersToLocal
```

```
            ' и удалить имя из коллекции.
            oName.Delete
        End If
    Next
End Sub
```

Объект ODBCError и коллекция ODBCErrors

В каждой коллекции ODBCErrors хранится список ошибок, которые возникли в результате последнего запроса через подключение ODBC. В каждом объекте ODBCError содержится информация об ошибке, которая возникла в результате последнего запроса через подключение ODBC. Если в результате последнего запроса через подключение ODBC не возникло ни одной ошибки, коллекция окажется пустой.

Кроме типичных атрибутов, коллекция ODBCErrors предоставляет свойство Count. Свойство Count возвращает количество объектов ODBCError, которые хранятся в коллекции.

Общие свойства ODBCError

Определения свойств Application, Creator и Parent приводились в начале приложения.

Свойства объекта ODBCError

Имя	*Тип возвращаемого значения*	*Описание*
ErrorString	String	Только чтение. Возвращает строку ошибки, сгенерированную подключением ODBC
SqlState	String	Только чтение. Возвращает ошибку состояния SQL, сгенерированную подключением ODBC

Пример: объект ODBCError и коллекция ODBCErrors

```
Sub CheckODBCErrors()
    Dim oErr As ODBCError
    Dim sMsg As String
    ' Продолжить выполнение после ошибок
    On Error Resume Next
    ' Не показывать запросы на регистрацию
    Application.DisplayAlerts = False
    ' Обновить таблицу запросов ODBC
    ActiveSheet.QueryTables(1).Refresh
    ' Возникли ошибки?
    If Application.ODBCErrors.Count = 0 Then
        ' Нет, значит все в порядке
        MsgBox "Обновление завершено"
    Else
        ' Да, поэтому перечислить ошибки
        sMsg = "Во время обновления произошли следующие ошибки"
        For Each oErr In Application.ODBCErrors
            sMsg = sMsg & vbCrLf & oErr.ErrorString & " (" & oErr.SqlState & ")"
        Next
        MsgBox sMsg
    End If
End Sub
```

Объект OLEDBError и коллекция OLEDBErrors

В коллекции OLEDBErrors хранится список ошибок, которые возникли в результате последнего запроса через поставщика OLE DB. В каждом объекте OLEDBError хранится информация о последней ошибке, которая возникла в результате последнего запроса через поставщика OLE DB. Если в результате последнего запроса через поставщика OLE DB не возникло ни одной ошибки, коллекция OLEDBErrors окажется пустой.

Кроме типичных атрибутов, коллекция OLEDBErrors предоставляет свойство Count. Это свойство возвращает количество объектов OLEDBError, которые хранятся в коллекции.

Общие свойства объекта OLEDBError

Определения свойств Application, Creator и Parent приводились в начале приложения.

Свойства объекта OLEDBError

Имя	*Тип возвращаемого значения*	*Описание*
ErrorString	String	Только чтение. Возвращает строку ошибки, сгенерированную поставщиком OLE DB
Native	Long	Только чтение. Возвращает специфичный для поставщика номер ошибки, характеризующий ошибку
Number	Long	Только чтение. Возвращает номер ошибки, характеризующий ошибку
SqlState	String	Только чтение. Возвращает ошибку состояния SQL, сгенерированную поставщиком OLE DB
Stage	Long	Только чтение. Возвращает стадию ошибки, сгенерированной поставщиком OLE DB

Пример: объект OLEDBError и коллекция OLEDBErrors

```
Sub CheckOLEDbErrors()
   Dim oErr As OLEDBError
   Dim sMsg As String
   ' Continue after errors
   On Error Resume Next
   ' Не выдавать запросов на регистрацию
   Application.DisplayAlerts = False
   ' Обновить сводную таблицу OLE DB
   ActiveSheet.PivotTables(1).Refresh
   ' Возникли ошибки?
   If Application.OLEDBErrors.Count = 0 Then
      ' Нет, все в порядке
      MsgBox "Обновление завершено"
   Else
      ' Да, поэтому перечислить ошибки
      sMsg = "В процессе обновления произошли следующие ошибки"
      For Each oErr In Application.OLEDBErrors
         sMsg = sMsg & vbCrLf & oErr.ErrorString & " (" & oErr.SqlState & ")"
      Next
      MsgBox sMsg
   End If
End Sub
```

Объект OLEFormat

В этом объекте хранятся все атрибуты объектов OLE или ActiveX, необходимые для связывания. Характеристики связывания описываются объектом LinkFormat. Родительским объектом в данном случае является объект Shape. Для использования этого объекта родительский объект Shape должен быть связан или внедрен.

Общие свойства объекта OLEFormat

Определения свойств Application, Creator и Parent приводились в начале приложения.

Свойства OLEFormat

Имя	*Тип возвращаемого значения*	*Описание*
Object	Object	Только чтение. Возвращает ссылку на родительский объект OLE
ProgId	String	Только чтение. Возвращает программный идентификатор родительского объекта OLE

Методы объекта OLEFormat

Имя	*Тип возвращаемого значения*	*Параметры*	*Описание*
Activate			Активизирует и открывает родительский объект OLE
Verb		[Verb]	Выполняет операцию над родительским объектом OLE. При этом вызывается реакция со стороны объекта OLE (например, xlOpen)

Пример: объект OLEFormat

```
Sub PrintEmbeddedWordDocuments1()
   Dim oShp As Shape
   Dim oOF As OLEFormat
   ' Перебрать все геометрические фигуры на листе
   For Each oShp In ActiveSheet.Shapes
      ' Является ли объект внедренным
      If oShp.Type = msoEmbeddedOLEObject Then
         ' Получить формат внедренного объекта
         Set oOF = oShp.OLEFormat

         ' Является ли этот объект документом Word?
         If oOF.ProgId Like "Word.Document*" Then

            ' Да, поэтому распечатать документ Word.
            ' Первый вызов .Object возвращает универсальный объект
            ' OLEObject, который хранится в объекте Shape.
            ' Второй вызов .Object возвращает объект Word,
            ' который хранится внутри OLEObject
            oOF.Object.Object.PrintOut
         End If
      End If
   Next
End Sub
```

Объект OLEObject и коллекция OLEObjects

В коллекции OLEObjects хранятся все элементы управления ActiveX, связанные объекты OLE и внедренные объекты OLE из листа или диаграммы. Объект OLEObject предоставляет элемент управления ActiveX, связанный объект OLE или внедренный объект OLE из листа или диаграммы.

Кроме типичных атрибутов, коллекция OLEObjects предоставляет множество свойств и методов. Эти свойства и методы перечислены в следующей таблице.

Свойства и методы коллекции OLEObjects

Имя	*Тип возвращаемого значения*	*Описание*
AutoLoad	Boolean	Устанавливает/возвращает необходимость автоматической загрузки объекта OLE при открытии книги. С объектами ActiveX это свойство не применяется. Обычно оно установлено в значение False. Используется, если в коллекции хранится только один объект OLEObject
Border	Border	Только чтение. Возвращает свойства границы вокруг объекта OLE. Используется, если в коллекции хранится только один объект OLEObject
Count	Long	Только чтение. Возвращает количество объектов OLEObject, которые хранятся в коллекции
Enabled	Boolean	Устанавливает/возвращает, включен ли объект OLEObject. Используется, если в коллекции хранится только один объект OLEObject
Height	Double	Устанавливает/возвращает высоту фрейма OLEObject. Используется, если в коллекции хранится только один объект OLEObject
Interior	Interior	Только чтение. Возвращает объект, содержащий параметры форматирования внутренней области объекта OLE (например, цвет внутренней области). Используется, если в коллекции хранится только один объект OLEObject
Left	Double	Устанавливает/возвращает расстояние от левого края фрейма объекта OLEObject до левого края листа. Используется, если в коллекции хранится только один объект OLEObject
Locked	Boolean	Устанавливает/возвращает возможность редактирования объекта при защите родительского листа. Используется, если в коллекции хранится только один объект OLEObject
Placement	Variant	Устанавливает/возвращает способ привязки объекта OLEObject к листу (например, произвольное размещение, перемещение вместе с ячейками). Для установки этого свойства можно воспользоваться константами XlPlacement. Используется, если в коллекции хранится только один объект OLEObject
PrintObject	Boolean	Устанавливает/возвращает необходимость печати объекта OLEObject при печати листа. Используется, если в коллекции хранится только один объект OLEObject

Имя	*Тип возвращаемого значения*	*Описание*
Shadow	Boolean	Устанавливает/возвращает необходимость отображения эффекта тени вокруг объекта OLE. Используется, если в коллекции хранится только один объект OLEObject
ShapeRange	ShapeRange	Только чтение. Возвращает объект OLE в виде объекта Shape. Используется, если в коллекции хранится только один объект OLEObject
SourceName	String	Устанавливает/возвращает имя источника ссылки для объекта OLE. Используется, если в коллекции хранится только один объект OLEObject
Top	Double	Устанавливает/возвращает расстояние от верхнего края объекта OLE до верхнего края родительского листа. Используется, если в коллекции хранится только один объект OLEObject
Visible	Boolean	Устанавливает/возвращает видимость всех объектов OLEObject в коллекции
Width	Double	Устанавливает/возвращает ширину фрейма объекта OLE. Используется, если в коллекции хранится только один объект OLEObject
ZOrder	Long	Только чтение. Возвращает положение объекта OLE относительно других объектов на листе. Используется, если в коллекции хранится только один объект OLEObject
Add	OLEObject	Метод. Параметры: [ClassType], [Filename], [Link], [DisplayAsIcon], [IconFileName], [IconIndex], [IconLabel], [Left], [Top], [Width], [Height]. Добавляет в коллекцию объект OLE. Положение нового объекта OLE определяется значением параметров Left, Top, Width и Height. Кроме этого, можно указать тип объекта OLEObject (параметр ClassType) и его расположение (параметр FileName). Остальные параметры являются аналогами существующих свойств объекта OLEObject
BringToFront	Variant	Метод. Перемещает все объекты OLE из коллекции на передний план поверх остальных объектов
Copy	Variant	Метод. Копирует все объекты OLE из коллекции в буфер обмена
CopyPicture	Variant	Метод. Параметры: Appearance As XlPictureAppearance, Format As XlCopyPictureFormat. Копирует объекты OLE из коллекции в буфер обмена в виде изображений. Параметр Appearance позволяет выбрать копирование экранного представления или представления для печати. Параметр Format позволяет выбрать тип изображения, которое копируется в буфер обмена
Cut	Variant	Метод. Вырезает все объекты OLE из коллекции в буфер обмена
Delete	Variant	Метод. Удаляет все объекты OLE из коллекции
Select	Variant	Метод. Параметры: [Replace]. Выделяет все объекты OLEObject из коллекции
SendToBack	Variant	Метод. Перемещает объект OLEObject на задний план за другими объектами из коллекции

Общие свойства объекра OLEObject

Определения свойств Application, Creator и Parent приводились в начале приложения.

Свойства объекта OLEObject

Имя	*Тип возвращаемого значения*	*Описание*
AltHTML	String	Устанавливает/возвращает тип кода HTML, который используется вместо встраивания элемента управления OLE при сохранении документа в виде Web-страницы
AutoLoad	Boolean	Устанавливает/возвращает необходимость автоматической загрузки объекта OLE при открытии книги. С объектами ActiveX это свойство не применяется. Обычно это свойство установлено в значение False
AutoUpdate	Boolean	Устанавливает/возвращает необходимость автоматического обновления объекта OLE при изменении исходного объекта. Применяется только со связанными объектами (OLEType=xlOLELink)
Border	Border	Только чтение. Возвращает свойства границы вокруг объекта OLE
Bottom-RightCell	Range	Только чтение. Возвращает диапазон, состоящий из одной ячейки и расположенный под левым нижним углом объекта OLE
Enabled	Boolean	Устанавливает/возвращает, включен ли объект OLEObject
Height	Double	Устанавливает/возвращает высоту фрейма объекта OLEObject
Index	Long	Только чтение. Возвращает положение текущего объекта OLEObject в коллекции
Interior	Interior	Только чтение. Возвращает объект с параметрами форматирования внутренней области объекта OLE (например, внутреннего цвета)
Left	Double	Устанавливает/возвращает расстояние от левого края фрейма объекта OLEObject до левого края листа
LinkedCell	String	Устанавливает/возвращает диапазон, в который записывается результат работы объекта OLE
ListFill-Range	String	Устанавливает/возвращает диапазон, в котором хранятся значения для списка ActiveX
Locked	Boolean	Устанавливает/возвращает возможность редактирования объекта в пределах защищенного листа
Name	String	Устанавливает/возвращает имя объекта OLE
Object		Только чтение. Предоставляет доступ к некоторым свойствам и методам объекта внутри объекта OLE
OLEType	Variant	Только чтение. Возвращает тип объекта OLE: xlOLELink или xlOLEEmbed. Для установки значения этого свойства можно воспользоваться константами XlOLEType

Имя	*Тип возвращаемого значения*	*Описание*
Placement	Variant	Устанавливает/возвращает тип привязки объекта OLEObject к листу (например, произвольное положение, перемещение вместе с ячейками). Для установки этого свойства можно воспользоваться константами XlPlacement
PrintObject	Boolean	Устанавливает/возвращает необходимость печати объекта OLEObject при печати листа
ProgId	String	Только чтение. Возвращает программный идентификатор, связанный с объектом OLE (например, "Excel.Application")
Shadow	Boolean	Устанавливает/возвращает необходимость отображения эффекта тени вокруг объекта OLE
ShapeRange	ShapeRange	Только чтение. Возвращает объект OLE в виде объекта Shape
SourceName	String	Устанавливает/возвращает исходное имя связи объекта OLE
Top	Double	Устанавливает/возвращает расстояние от верхнего края объекта OLE до верхнего края родительского листа
TopLeftCell	Range	Только чтение. Возвращает диапазон, состоящий из единственной ячейки и расположенный над левым верхним углом объекта OLE
Visible	Boolean	Устанавливает/возвращает необходимость отображения объекта OLEObject на экране
Width	Double	Устанавливает/возвращает ширину фрейма объекта OLE
ZOrder	Long	Только чтение. Возвращает положение объекта OLE относительно других объектов на листе

Методы объекта OLEObject

Имя	*Тип возвращаемого значения*	*Параметры*	*Описание*
Activate	Variant		Устанавливает фокус и активизирует объект OLE
BringToFront	Variant		Размещает объект OLE на переднем плане поверх остальных объектов
Copy	Variant		Копирует объект OLE в буфер обмена
CopyPicture	Variant	Appearance As XlPicture Appearance, Format As XlCopyPicture Format	Копирует изображение объекта OLE в буфер обмена. Параметр Appearance позволяет выбрать копирование экранного представления или представления для печати. Параметр Format позволяет указать тип изображения, которое будет скопировано в буфер обмена
Cut	Variant		Вырезает объект OLE в буфер обмена
Delete	Variant		Удаляет объект OLEObject
Duplicate			Дублирует объект OLEObject на родительском листе
Select	Variant	[Replace]	Выделяет объект OLEObject

Имя	*Тип возвращаемого значения*	*Параметры*	*Описание*
SendToBack	Variant		Размещает объект OLEObject позади остальных объектов
Update	Variant		Обновляет связь объекта OLE (если такая связь существует)
Verb	Variant	Verb As XlOLEVerb	Вызывает операцию родительского объекта OLE, которая провоцирует реакцию объекта OLE (например, xlOpen)

События объекта OLEObject

Имя	*Параметры*	*Описание*
GotFocus		Возникает при получении фокуса объектом OLE
LostFocus		Возникает при потере фокуса объектом OLE

Пример: объект OLEObject и коллекция OLEObjects

```
Sub PrintEmbeddedWordDocuments2()
   Dim oOLE As OLEObject
   ' Перебрать все геометрические фигуры на листе
   For Each oOLE In ActiveSheet.OLEObjects
      ' Это документ Word?
      If oOLE.ProgId Like "Word.Document*" Then
         ' Да, поэтому напечатать документ Word.
         oOLE.Object.PrintOut
      End If
   Next
End Sub
```

Объект Outline

Объект Outline предоставляет возможность управления структурой Excel. Родительским для объекта Outline является объект Worksheet.

Общие свойства объекта Outline

Определения свойств Application, Creator и Parent приводились в начале приложения.

Свойства объекта Outline

Имя	*Тип возвращаемого значения*	*Описание*
AutomaticStyles	Boolean	Устанавливает/возвращает необходимость автоматического назначения стилей
SummaryColumn	XlSummaryColumn	Устанавливает/возвращает расположение итогового столбца: справа (xlRight) или слева (xlLeft)
SummaryRow	XlSummaryRow	Устанавливает/возвращает расположение итогового столбца: выше (xlAbove) или ниже (xlBelow)

Методы объекта Outline

Имя	Тип возвращаемого значения	Параметры	Описание
ShowLevel	Variant	[RowLevels], [ColumnLevels]	Показывает информацию о строках и столбцах более высокого уровня, на которые указывают параметры RowLevels и ColumnLevels соответственно. Информация об остальных уровнях остается скрытой

Пример: объект Outline

```
Sub ShowOutlines()
   Dim oOutl As Outline
   ' Сгруппировать некоторые строки
   ActiveSheet.Range("4:5").Group
   ' Получить объект Outline
   Set oOutl = ActiveSheet.Outline
   ' Отформатировать отображение структуры
   With oOutl
      .ShowLevels 1
      .SummaryRow = xlSummaryAbove
   End With
End Sub
```

Объект PageSetup

Объект PageSetup предоставляет доступ к функциональности диалогового окна Параметры страницы (Page Setup). Возможными родительскими объектами могут быть объект Chart и объект Worksheet.

Общие свойства объекта PageSetup

Определения свойств Application, Creator и Parent приводились в начале приложения.

Свойства объекта PageSetup

Имя	Тип возвращаемого значения	Описание
BlackAndWhite	Boolean	Устанавливает/возвращает необходимость печати элементов листа в черно-белом режиме. Если родительским объектом является Chart, это свойство не применяется
BottomMargin	Double	Устанавливает/возвращает нижнее поле страницы в пунктах
CenterFooter	String	Устанавливает/возвращает текст центральной части верхнего колонтитула
CenterFooterPicture	Graphic	Только чтение. Возвращает изображение, которое расположено в центральной части колонтитула
CenterHorizontally	Boolean	Устанавливает/возвращает необходимость расположения листа или диаграммы в центре листа по горизонтали
CenterVertically	Boolean	Устанавливает/возвращает необходимость расположения листа или диаграммы в центре листа по вертикали

Имя	*Тип возвращаемого значения*	*Описание*
ChartSize	XlObjectSize	Устанавливает/возвращает необходимость масштабирования диаграммы для размещения на одной странице (xlFullPage или xlFitToPage) или печати с теми же размерами, что и на экране (xlScreenSize). Это свойство не применяется, если родительским объектом является Worksheet
Draft	Boolean	Устанавливает/возвращает необходимость печати графических элементов. Значение True означает, что графические изображения печататься не будут
FirstPage-Number	Long	Устанавливает/возвращает номер первой страницы. Для автоматической установки номера воспользуйтесь константой xlAutomatic (это поведение, принятое по умолчанию)
FitToPages-Tall	Variant	Устанавливает/возвращает количество страниц в высоту, на которые будет распечатан лист. Если установить это свойство в значение False, будет использовано свойство FitToPagesWide
FitToPages-Wide	Variant	Устанавливает/возвращает количество страниц в ширину, на которые будет распечатан лист. Если установить это свойство в значение False, будет использовано свойство FitToPagesTall
FooterMargin	Double	Устанавливает/возвращает расстояние от нижнего края страницы до нижнего колонтитула (в пунктах)
HeaderMargin	Double	Устанавливает/возвращает расстояние от верхнего края страницы до верхнего колонтитула (в пунктах)
LeftFooter	String	Устанавливает/возвращает текст левой части нижнего колонтитула
LeftFooter-Picture	Graphic	Только чтение. Возвращает изображение для левой части нижнего колонтитула
LeftHeader	String	Устанавливает/возвращает текст левой части верхнего колонтитула
LeftHeader-Picture	Graphic	Только чтение. Возвращает изображение для левой части верхнего колонтитула
LeftMargin	Double	Устанавливает/возвращает левое поле страницы в пунктах
Order	XlOrder	Устанавливает/возвращает способ нумерации страниц при печати больших листов (например, xlDownTheOVer, xlOverThenDown). Это свойство не используется, если родительским объектом является Chart
Orientation	XlPage-Orientation	Устанавливает/возвращает ориентацию страницы: xlLandscape или xlPortrait
PaperSize	XlPaperSize	Устанавливает/возвращает размер бумаги (например, xlPaperLetter, xlPaperLegal)
PrintArea	String	Устанавливает/возвращает печатаемый диапазон листа. Если это свойство установлено в значение False, будет напечатан весь лист. Это свойство не используется, если родительским объектом является объект Chart

Имя	*Тип возвращаемого значения*	*Описание*
`PrintComments`	`XlPrintErrors`	Устанавливает/возвращает тип отображаемых ошибок печати. Это позволяет подавлять сообщения об ошибках при печати листа
`PrintGridlines`	`Boolean`	Устанавливает/возвращает необходимость печати направляющих линий листа. Не используется, если родительским объектом является `Chart`
`PrintHeadings`	`Boolean`	Устанавливает/возвращает необходимость печати заголовков столбцов и строк
`PrintNotes`	`Boolean`	Устанавливает/возвращает необходимость печати связанных с ячейками заметок. Не используется, если родительским объектом является `Chart`
`PrintTitleColumns`	`String`	Устанавливает/возвращает столбцы, которые необходимо повторять в левой части каждой страницы
`PrintTitleRows`	`String`	Устанавливает/возвращает строки, которые необходимо повторять в верхней части каждой страницы
`RightFooter`	`String`	Устанавливает/возвращает текст правой части нижнего колонтитула
`RightFooterPicture`	`Graphic`	Только чтение. Возвращает изображение для правой части нижнего колонтитула
`RightHeader`	`String`	Устанавливает/возвращает текст правой части верхнего колонтитула
`RightHeaderPicture`	`Graphic`	Только чтение. Возвращает изображение для правой части верхнего колонтитула
`RightMargin`	`Double`	Устанавливает/возвращает размер правого поля страницы в пунктах
`TopMargin`	`Double`	Устанавливает/возвращает размер верхнего поля страницы в пунктах
`Zoom`	`Variant`	Устанавливает/возвращает степень масштабирования листа (в процентах от 10 до 400). Не используется, если родительским объектом является `Chart`

Методы объекта PageSetup

Имя	*Тип возвращаемого значения*	*Параметры*	*Описание*
`PrintQuality`	`Variant`	`[Index]`	Устанавливает/возвращает качество печати. Параметр `Index` позволяет указать качество печати по горизонтали (1) и вертикали (2)

Пример: объект PageSetup

```
Sub SetUpPage()
   Dim oPS As PageSetup
   ' Получить объект PageSetup для листа
   Set oPS = ActiveSheet.PageSetup
   ' Настроить страницу
   With oPS
      ' Установить размер страницы в соответствии с
      ' принятыми в данной местности требованиями.
```

```
        .PaperSize = fnLocalPaperSize
        .Orientation = xlPortrait
        ' и т.д.
    End With
End Sub
Function fnLocalPaperSize() As XlPaperSize
    ' Сохранить размер листа при чтении
    Static iPaperSize As XlPaperSize
    ' Размер установлен?
    If iPaperSize = 0 Then
        ' Нет, поэтому создать новую книгу и считать размер листа
        With Workbooks.Add
            iPaperSize = .Worksheets(1).PageSetup.PaperSize
            .Close False
        End With
    End If
    ' Вернуть размер листа
    fnLocalPaperSize = iPaperSize
End Function
```

Объект Pane и коллекция Panes

Коллекция Panes поддерживает управление различными панелями окна. Объект Pane является эквивалентом единственной панели окна. Родительским для коллекции Panes является объект Window.

Кроме типичных атрибутов, коллекция Panes предоставляет свойство Count. Это свойство возвращает количество объектов Pane в коллекции.

Общие свойства коллекции Pane

Определения свойств Application, Creator и Parent приводились в начале приложения.

Свойства объекта Pane

Имя	*Тип возвращаемого значения*	*Описание*
Index	Long	Только чтение. Возвращает положение объекта Pane в коллекции
ScrollColumn	Long	Устанавливает/возвращает номер самого левого столбца в панели окна
ScrollRow	Long	Устанавливает/возвращает номер самой верхней строки в панели окна
VisibleRange	Range	Только чтение. Возвращает диапазон ячеек, видимых в панели окна

Методы объекта Pane

Имя	*Тип возвращаемого значения*	*Параметры*	*Описание*
Activate	Boolean		Активизирует панель
LangeScroll	Variant	[Down], [Up], [ToRight], [ToLeft]	Выполняет прокрутку документа в указанном направлении на целый экран

Имя	*Тип возвращаемого значения*	*Параметры*	*Описание*
ScrollIntoView		Left As Long, Top As Long, Width As Long, Height As Long, [Start]	Прокручивает документ для отображения точки, которая определяется параметрами Left, Top, Width и Height. Указанная точка располагается в верхнем левом углу панели (Start = True) или в нижнем правом углу панели (Start = False). Параметры Left, Top, Width и Height указываются в пунктах
SmallScroll	Variant	[Down], [Up], [ToRight], [ToLeft]	Прокручивает документ в указанном направлении на одну строку

Пример: объект Pane и коллекция Panes

```
Sub ScrollActivePane()
   Dim oPane As Pane
   Dim oRNg As Range
   ' Диапазон, который будет показан в панели
   Set oRNg = Range("G3:J10")
   ' Получить активную панель
   Set oPane = Application.ActiveWindow.ActivePane
   ' Прокрутить панель для отображения диапазона в левом верхнем углу
   oPane.ScrollColumn = oRNg.Column
   oPane.ScrollRow = oRNg.Row
End Sub
```

Объект Parameter и коллекция Parameters

В коллекции Parameters хранится список параметров, связанных с таблицей запросов. Если ни одного параметра не существует, в коллекции отсутствуют объекты Parameter. Каждый объект Parameter соответствует одному параметру таблицы запроса. Родительским для коллекции Parameters является объект QueryTable.

Кроме типичных атрибутов, коллекция Parameters предоставляет несколько дополнительных свойств. Все они перечислены в следующей таблице.

Свойства и методы коллекции Parameters

Имя	*Тип возвращаемого значения*	*Описание*
Count	Long	Только чтение. Возвращает количество объектов Parameters в коллекции
Add	Parameter	Метод. Параметры: Name As String, [iDataType]. Добавляет параметр в коллекцию, создавая новый параметр запроса для родительской таблицы запроса. Тип создаваемого параметра указывается через параметр iDataType (для этого можно воспользоваться константами XlParameterDataType)
Delete		Метод. Удаляет объекты Parameter из коллекции

Общие свойства объекта Parameter

Определения свойств Application, Creator и Parent приводились в начале приложения.

Свойства объекта Parameter

Имя	*Тип возвращаемого значения*	*Описание*
DataType	XlParameter-DataType	Устанавливает/возвращает тип данных параметра
Name	String	Устанавливает/возвращает имя параметра
PromptString	String	Только чтение. Возвращает вопрос, который задается пользователю для получения значения параметра
RefreshOn-Change	Boolean	Устанавливает/возвращает необходимость обновления результатов таблицы запроса при изменении значения параметра
SourceRange	Range	Только чтение. Возвращает диапазон, в котором хранится значение параметра
Type	XlParameter-Type	Только чтение. Возвращает тип параметра (например, xlConstant, xlPrompt или xlRange). xlConstant означает, что значение параметра хранится в свойстве Value. xlPrompt означает, что значение запрашивается у пользователя. xlRange означает, что в свойстве Value определяется диапазон ячеек, в которых хранится значение параметра
Value	Variant	Только чтение. Возвращает значение параметра

Методы объекта Parameter

Имя	*Тип возвращаемого значения*	*Параметры*	*Описание*
SetParam		Type As XlParameterType, Value	Устанавливает/возвращает тип параметра и значение параметра

Пример: объект Parameter и коллекция Parameters

```
Sub UpdateQuery()
   Dim oParam As Parameter
   ' Используется таблица запроса?
   With ActiveSheet.QueryTables(1)
      ' Получить первый параметр
      Set oParam = .Parameters(1)

      ' Установить его значение
      oParam.SetParam xlConstant, "Company"

      ' Обновить запрос
      .Refresh
   End With
End Sub
```

Объект Phonetic и коллекция Phonetics

В коллекции Phonetics хранится весь фонетический текст диапазона. Объект Phonetic соответствует единственной строке фонетического текста. Родительским для коллекции Phonetics является объект Range.

Кроме типичных атрибутов, коллекция Phonetics предоставляет несколько свойств и методов. Они перечислены в следующей таблице.

Свойства и методы коллекции Phonetics

Имя	*Тип возвращаемого значения*	*Описание*
Alignment	Long	Устанавливает/возвращает выравнивание фонетического текста. Для установки этого свойства используются константы XlPhoneticAignment
Character-Type	Long	Устанавливает/возвращает тип используемого фонетического текста. Для установки этого свойства используются константы XlPhoneticCharacterType
Count	Long	Только чтение. Возвращает количество объектов Phonetic, хранящихся в коллекции
Font	Font	Только чтение. Возвращает объект, в котором хранятся параметры шрифта для текста в коллекции Phonetics
Length	Long	Только чтение. Возвращает количество символов фонетического текста начиная с точки, определенной свойством Start
Start	Long	Только чтение. Возвращает положение первого символа строки фонетического текста. Свойство используется, только если в коллекции существуют объекты Phonetic
Text	String	Устанавливает/возвращает фонетический текст. Свойство используется, только если в коллекции существуют объекты Phonetic
Visible	Boolean	Устанавливает/возвращает видимость фонетического текста для пользователя. Свойство применяется, только если в коллекции существуют объекты Phonetic
Add		Метод. Параметры: Start As Long, Length As Long, Text As String. Добавляет в коллекцию объект Phonetic, расположенный в ячейке родительского объекта Range
Delete		Метод. Удаляет все объекты Phonetic из коллекции

Общие свойства объекта Phonetic

Определения свойств Application, Creator и Parent приводились в начале приложения.

Свойства объекта Phonetic

Имя	*Тип возвращаемого значения*	*Описание*
Alignment	Long	Устанавливает/возвращает выравнивание фонетического текста. Для установки свойства используются константы XlPhoneticAlignment
CharacterType	Long	Устанавливает/возвращает тип используемого фонетического текста. Для установки свойства применяются константы XlPhoneticCharacterType
Font	Font	Только чтение. Возвращает объект, содержащий параметры шрифта для фонетического текста
Text	String	Устанавливает/возвращает фонетический текст
Visible	Boolean	Устанавливает/возвращает видимость фонетического текста для конечного пользователя

Объект PictureFormat

Этот объект предоставляет возможность управления свойствами рисунка в пределах родительского объекта Shape.

Общие свойства объекта PictureFormat

Определения свойств Application, Creator и Parent приводились в начале приложения.

Свойства объекта PictureFormat

Имя	*Тип возвращаемого значения*	*Описание*
Brightness	Single	Устанавливает/возвращает яркость родительской геометрической фигуры (от 0 до 1, где 1 соответствует максимальной яркости)
ColorType	MsoPictureColorType	Устанавливает/возвращает тип цвета родительской геометрической фигуры
Contrast	Single	Устанавливает/возвращает контраст родительской геометрической фигуры (от 0 до 1, где 1 соответствует максимальному контрасту)
CropBottom	Single	Устанавливает/возвращает размер удаляемого фрагмента в нижней части рисунка
CropLeft	Single	Устанавливает/возвращает размер удаляемого фрагмента в левой части рисунка
CropRight	Single	Устанавливает/возвращает размер удаляемого фрагмента в правой части рисунка
CropTop	Single	Устанавливает/возвращает размер удаляемого фрагмента в верхней части рисунка
TransparencyColor	Long	Устанавливает/возвращает цвет, который используется для обозначения прозрачности
TransparentBackground	MsoTriState	Устанавливает/возвращает необходимость включения прозрачности для прозрачного цвета

Методы объекта PictureFormat

Имя	*Тип возвращаемого значения*	*Параметры*	*Описание*
`IncrementBrightness`		`Increment As Single`	Увеличивает яркость на значение параметра `Increment`
`IncrementContrast`		`Increment As Single`	Увеличивает контраст на значение параметра `Increment`

Пример: объект PictureFormat

```
Sub SetPictureFormat()
   Dim oShp As Shape
   Dim oPF As PictureFormat
   For Each oShp In ActiveSheet.Shapes
      If oShp.Type = msoPicture Then

            ' Получить объект PictureFormat
         Set oPF = oShp.PictureFormat
         ' Установить формат рисунка

         With oPF
            .TransparentBackground = msoTrue
            .TransparencyColor = RGB(255, 0, 0)
            .ColorType = msoPictureWatermark
         End With
      End If
   Next
End Sub
```

Объект PivotCache и коллекция PivotCaches

В коллекции `PivotCaches` хранятся области памяти с данными отчетов сводной таблицы. Каждый объект `PivotCache` соответствует одному фрагменту памяти для отчета сводной таблицы. Родительским объектом коллекции `PivotCaches` является объект `Workbook`. Кроме этого, для объекта `PivotCache` родительским может быть объект `PivotTable`.

Кроме типичных атрибутов, коллекция `PivotCaches` предоставляет свойство `Count` и метод `Add`. Свойство `Count` возвращает количество объектов в коллекции. Метод `Add` принимает параметр `SourceType` (значением этого параметра может быть одна из констант `XlPivotTableSourceType`) и параметр `SourceData` и добавляет объект `PivotCache` в коллекцию.

Общие свойства объекта PivotCache

Определения свойств `Application`, `Creator` и `Parent` приводились в начале приложения.

Свойства объекта PivotCache

Имя	*Тип возвращаемого значения*	*Описание*
`ADOConnection`	`Object`	Только чтение. Возвращает объект подключения ADO, если объект `PivotCache` подключается к источнику данных OLE DB

Имя	*Тип возвращаемого значения*	*Описание*
BackgroundQuery	Boolean	Устанавливает/возвращает необходимость асинхронной обработки запросов в отчете сводной таблицы. Значение False используется для источников данных OLAP
CommandText	Variant	Устанавливает/возвращает команду SQL, которая используется для получения данных
CommandType	XlCmdType	Устанавливает/возвращает тип свойства CommandText (например, xlCmdSQL, xlCmdTable)
Connection	Variant	Устанавливает/возвращает строку подключения OLE DB, строку ODBC, источник данных в Web, путь к текстовому файлу или путь к базе данных
EnableRefresh	Boolean	Устанавливает/возвращает возможность обновления данных в кэше сводной таблицы. Для источников данных OLAP это свойство всегда установлено в значение False
Index	Long	Только чтение. Возвращает положение объекта в коллекции
IsConnected	Boolean	Только чтение. Возвращает состояние подключения объекта PivotCache к источнику данных
LocalConnection	String	Устанавливает/возвращает строку подключения для автономного файла куба. Для источников данных, отличных от OLAP, это свойство не устанавливается. Используется совместно со свойством UseLocalConnection
MaintainConnection	Boolean	Устанавливает/возвращает необходимость сохранения подключения к источнику данных до закрытия книги. Используется только для источников данных OLE DB
MemoryUsed	Long	Только чтение. Возвращает объем памяти (в байтах), выделенной для объекта PivotCache
MissingItemsLimit	XlPivotTableMissingItems	Устанавливает/возвращает максимальное количество уникальных элементов, которые сохраняются для поля сводной таблицы, даже если в записях кэша отсутствуют соответствующие данные
OLAP	Boolean	Только чтение. Возвращает состояние подключения объекта PivotCache к серверу OLAP
OptimizeCache	Boolean	Устанавливает/возвращает необходимость оптимизации кэша сводной таблицы при построении. Для источников данных OLE DB это свойство всегда установлено в значение False
QueryType	xlQueryType	Только чтение. Возвращает тип подключения, связанного с таблицей запроса (например, xlOLEDBQuery, xlDAOQuery, xlTextImport)
RecordCount	Long	Только чтение. Возвращает количество записей в кэше сводной таблицы
Recordset		Устанавливает/возвращает набор записей, который используется в качестве источника данных для кэша сводной таблицы
RefreshDate	Date	Только чтение. Возвращает дату последнего обновления кэша

Имя	*Тип возвращаемого значения*	*Описание*
RefreshOn-FileOpen	Boolean	Устанавливает/возвращает необходимость обновления кэша сводной таблицы при открытии книги
Refresh-Period	Long	Устанавливает/возвращает длительность интервала (в минутах) автоматического обновления из источника данных. Для отключения автоматического обновления установите свойство в значение 0
Robust-Connect	XlRobust-Connect	Устанавливает/возвращает метод, с помощью которого объект PivotCache подключается к источнику данных
Save-Password	Boolean	Устанавливает/возвращает необходимость сохранения пароля подключения ODBC в таблице запроса
Source-Connection-File	String	Устанавливает/возвращает имя файла, который использовался для создания сводной таблицы
SourceData	Variant	Устанавливает/возвращает источник данных для отчета сводной таблицы
SourceData-File	String	Только чтение. Возвращает имя файла с исходными данными для объекта PivotCache
SourceType	XlPivotTable-SourceType	Только чтение. Возвращает значение, идентифицирующее тип публикуемого элемента
UseLocal-Connection	Boolean	Устанавливает/возвращает необходимость применения свойства LocalConnection для установки источника данных. Значение False означает, что используется свойство Connection. Это свойство позволяет автономно работать с некоторыми источниками данных

Методы объекта PivotCache

Имя	*Тип возвращаемого значения*	*Параметры*	*Описание*
CreatePivot-Table	PivotTable	TableDestination As Variant, [TableName], [ReadData], [DefaultVersion]	Создает отчет сводной таблицы, который основан на текущем объекте PivotCache. Параметр TableDestination указывает на расположение нового отчета сводной таблицы. Кроме этого, можно указать значение параметра TableName. Для заполнения кэша всеми записями из внешней базы данных параметр ReadData устанавливается в значение True. Для получения только части данных параметр ReadData устанавливается в значение False. Параметр DefaultVersion позволяет указать принятую по умолчанию версию отчета сводной таблицы
Make-Connection			Устанавливает подключение для указанного объекта PivotCache

Имя	*Тип возвращаемого значения*	*Параметры*	*Описание*
Refresh			Обновляет данные в кэше сводной таблицы из последней копии внешних данных. Для асинхронного обновления данных параметр BackgroundQuery устанавливается в значение True
ResetTimer			Сбрасывает таймер автоматического обновления, который устанавливается с помощью свойства RefreshPeriod
SaveAsODC		ODCFileName As String, [Description], [Keywords]	Сохраняет источник объекта PivotCache в файле Office Data Connection (ODC). Параметр ODCFileName указывает на положение сохраняемого файла. Параметр Description содержит описание содержимого файла. Через параметр Keywords передается разделенный пробелами список ключевых слов, которые можно использовать при поиске этого файла

Пример: объект PivotCache и коллекция PivotCaches

```
Sub RefreshPivotCache()
    Dim oPC As PivotCache
   Set oPC = ActiveWorkbook.PivotCaches(1)
   With oPC
      ' Обновление в интерактивном режиме
      .BackgroundQuery = False

      ' Обновлять только те данные, которые старше 1 часа
      If .RefreshDate < Now - TimeValue("01:00:00") Then
         .Refresh
      End If
   End With
End Sub
```

Объект PivotCell

Этот объект соответствует ячейке в пределах сводной таблицы. Для доступа к объекту PivotCell используется объект Range. После этого свойства объекта PivotCell применяются для извлечения данных из сводной таблицы. Например, свойства PivotCellType, ColumnItems и RowItems позволяют получить объем продаж, сделанных конкретным продавцом по заданному региону.

Функциональность этого объекта дублирует функциональность функции листа GETPIVOTDATA и метода PivotTable.GetPivotData. Но при этом объект PivotCell предоставляет информацию о положении ячейки в отчете. Функция GETPIVOTDATA и метод GetPivotData обеспечивают обратную функциональность. Они возвращают значение, связанное с предоставленными заголовками строки и столбца.

Общие свойства объекта PivotCell

Определения свойств Application, Creator и Parent приводились в начале приложения.

Свойства объекта PivotCell

Имя	*Тип возвращаемого значения*	*Описание*
ColumnItems	PivotItemList	Только чтение. Возвращает элементы, которые соответствуют выделенному диапазону на оси столбцов
CustomSubtotal-Function	XlConsolidation-Function	Только чтение. Возвращает собственную функцию итогов для объекта PivotCell
DataField	PivotField	Только чтение. Возвращает выделенное поле данных
PivotCellType	XlPivotCellType	Только чтение. Возвращает сущность сводной таблицы, которой соответствует выделенная ячейка
PivotField	PivotField	Только чтение. Возвращает поле сводной таблицы, содержащее верхний левый угол указанного диапазона
PivotItem	PivotItem	Только чтение. Возвращает элемент сводной таблицы, содержащий верхний левый угол указанного диапазона
PivotTable	PivotTable	Только чтение. Возвращает отчет сводной таблицы, в котором содержится верхний левый угол указанного диапазона, или отчет сводной таблицы, связанный с отчетом сводной диаграммы
Range	Range	Только чтение. Возвращает диапазон, к которому относится указанный объект PivotCell
RowItems	PivotItemList	Только чтение. Возвращает элементы, представляющие выделенный диапазон на оси строк

Объект PivotField, коллекция PivotFields и коллекция CalculatedFields

В коллекции PivotFields хранятся поля, связанные с родительским отчетом сводной таблицы. В коллекции CalculatedFields хранятся вычисляемые поля, связанные с родительским отчетом сводной таблицы. Каждый объект PivotField соответствует одному полю в отчете сводной таблицы. Родительским объектом для коллекций PivotFields и CalculatedFields является объект PivotTable.

Кроме типичных атрибутов, коллекции PivotFields и CalculatedFields предоставляют одно дополнительное свойство Count, возвращающее количество полей в родительской коллекции. Кроме этого, коллекция CalculatedFields предоставляет метод Add, который добавляет в коллекцию новое вычислимое поле с использованием параметров Name и Formula.

Общие свойства объекта PivotField

Определения свойств Application, Creator и Parent приводились в начале приложения.

Свойства объекта PivotField

Имя	*Тип возвращаемого значения*	*Описание*
AutoShow-Count	Long	Только чтение. Возвращает количество верхних или нижних элементов, которые автоматически отображаются в поле сводной таблицы
AutoShow-Field	String	Только чтение. Возвращает имя поля с данными, которое используется для определения списка верхних или нижних элементов, отображаемых автоматически в поле сводной таблицы
AutoShow-Range	Long	Только чтение. Возвращает значение xlTop, если автоматически отображаются верхние элементы, или xlBottom, если отображаются нижние элементы
AutoShow-Type	Long	Только чтение. Возвращает xlAutomatic, если свойство AutoShow установлено в значение True, или xlManual, если свойство AutoShow отключено
AutoSoft-Field	String	Только чтение. Возвращает имя поля данных, которое используется для автоматической сортировки полей сводной таблицы
AutoSort-Order	Long	Только чтение. Возвращает одну из констант XLSortOrder, описывающих тип автоматического порядка сортировки полей
BaseField	Variant	Устанавливает/возвращает базовое поле, которое используется для расчетов. Относится только к полям данных
BaseItem	Variant	Устанавливает/возвращает базовый элемент в базовом поле, который используется для расчетов. Относится только к полям данных
Calculation	XlPivotField-Calculation	Устанавливает/возвращает тип расчетов, которые выполняются по отношению к полю данных
Caption	String	Устанавливает/возвращает текстовую метку, которая используются для обозначения поля
ChildField	PivotField	Только чтение. Возвращает дочернее поле текущего поля (если такое поле существует)
ChildItems	Variant	Только чтение. Параметры: [Index]. Возвращает объект или коллекцию, содержащую один (PivotItem) или группу элементов сводной таблицы (PivotItems), связанных с полем
CubeField	CubeField	Только чтение. Возвращает поле куба, на основе которого создавалось текущее поле сводной таблицы
CurrentPage		Устанавливает/возвращает текущую страницу, на которой показано поле страницы. Относится только к полям страниц
Current-PageList	Variant	Устанавливает/возвращает массив строк, которые соответствуют списку элементов, входящих в многоэлементное поле страницы отчета сводной таблицы
Current-PageName	String	Устанавливает/возвращает отображаемую страницу отчета сводной таблицы
DataRange	Range	Только чтение. Возвращает диапазон, содержащий данные или элементы поля

Имя	*Тип возвращаемого значения*	*Описание*
DataType	XlPivotField-DataType	Только чтение. Возвращает тип данных поля сводной таблицы
DatabaseSort	Boolean	Устанавливает/возвращает возможность ручного перемещения элементов внутри поля сводной таблицы. Если в поле отсутствуют перемещенные вручную элементы, это свойство возвращает значение True
DragToColumn	Boolean	Устанавливает/возвращает возможность перетаскивания поля в позицию столбца
DragToData	Boolean	Устанавливает/возвращает возможность перетаскивания поля в позицию данных
DragToHide	Boolean	Устанавливает/возвращает возможность перетаскивания поля за пределы отчета сводной таблицы с последующим сокрытием
DragToPage	Boolean	Устанавливает/возвращает возможность перетаскивания поля в позицию страницы
DragToRow	Boolean	Устанавливает/возвращает возможность перетаскивания поля в позицию строки
DrilledDown	Boolean	Устанавливает/возвращает возможность разворачивания поля
EnableItem-Selection	Boolean	Устанавливает/возвращает возможность использования раскрывающегося списка поля через пользовательский интерфейс
Formula	String	Устанавливает/возвращает формулу, связанную с полем сводной таблицы
Function	XlConsolida-tionFunction	Устанавливает/возвращает тип функции, которая используется для получения итогов поля сводной таблицы
GroupLevel	Variant	Только чтение. Возвращает размещение поля в группе полей
HiddenItems	Variant	Только чтение. Параметры: [Index]. Возвращает объект или коллекцию, содержащие один (PivotItem) или группу (PivotItems) скрытых элементов сводной таблицы, связанных с полем
HiddenItems-List	Variant	Устанавливает/возвращает массив строк, которые являются скрытыми элементами поля сводной таблицы
IsCalculated	Boolean	Только чтение. Возвращает информацию о том, является ли поле сводной таблицы вычислимым
IsMember-Property	Boolean	Только чтение. Возвращает наличие свойств внутри поля сводной таблицы
LabelRange	Range	Только чтение. Возвращает диапазон ячеек, содержащий метку поля
Layout-BlankLine	Boolean	Устанавливает/возвращает необходимость добавления пустой строки после текущей строки поля
LayoutForm	XlLayoutForm-Type	Устанавливает/возвращает способ отображения элементов в пределах поля
LayoutPage-Break	Boolean	Устанавливает/возвращает необходимость вставки разрыва страницы после каждого поля

Имя	*Тип возвращаемого значения*	*Описание*
LayoutSubtotalLocation	XlSubtotalLocationType	Устанавливает/возвращает расположение итогов полей относительно текущего поля
MemoryUsed	Long	Только чтение. Возвращает выделенную для хранения поля память в байтах
Name	String	Устанавливает/возвращает имя поля
NumberFormat	String	Устанавливает/возвращает формат, который используется для отображения чисел
Orientation	XlPivotFieldOrientation	Устанавливает/возвращает расположение поля в отчете сводной таблицы
ParentField	PivotField	Только чтение. Возвращает родительское поле текущего поля
ParentItems	Variant	Только чтение. Параметры: [Index]. Возвращает объект или коллекцию, содержащую один (PivotItem) или группу (PivotItems) родительских элементов сводной таблицы, связанных с полем
Position	Variant	Устанавливает/возвращает положение поля среди других полей с такой же ориентацией
PropertyOrder	Long	Устанавливает/возвращает положение свойства на экране внутри поля куба, которому свойство принадлежит (установка этого свойства приведет к изменению порядка). Используется только вместе с объектами PivotField, которые являются полями свойств-членов
PropertyParentField	PivotField	Только чтение. Возвращает поле, с которым связаны свойства в этом поле
ServerBased	Boolean	Устанавливает/возвращает необходимость извлечения из внешнего источника данных только тех элементов, которые соответствуют выделенным полям страницы
ShowAllItems	Boolean	Устанавливает/возвращает необходимость отображения всех элементов поля
SourceName	String	Только чтение. Возвращает имя источника данных для этого поля
StandardFormula	String	Устанавливает/возвращает формулы с использованием региональных стандартов США
SubtotalName	String	Устанавливает/возвращает метку, которая используется для столбца или строки итогов для этого поля
Subtotals	Variant	Параметры: [Index]. Устанавливает/возвращает итоги, показанные для этого поля
TotalLevels	Variant	Только чтение. Возвращает общее количество полей в текущей группе полей
Value	String	Устанавливает возвращает имя поля
VisibleItems	Variant	Только чтение. Параметры: [Index]. Возвращает объект или коллекцию, содержащую один (PivotItem) или группу (PivotItems) видимых элементов сводной таблицы, связанных с полем

Методы объекта PivotField

Имя	Тип возвращаемого значения	Параметры	Описание
AddPageItem		Item As String, [ClearList]	Добавляет дополнительные элементы в поле страницы, состоящее из нескольких элементов. Параметр Item содержит имя исходного объекта PivotItem, соответствующего уникальному имени конкретного члена OLAP. Параметр ClearList указывает на необходимость удаления существующих элементов перед добавлением нового элемента
AutoShow		Type As Long, Range As Long, Count As Long, Field As String	Устанавливает количество верхних или нижних элементов, которые должны отображаться для поля строки, страницы или столбца. Параметр Type описывает автоматическое или ручное расположение элементов (xlAutomatic или xlManual). Параметр Range указывает на расположение начала списка элементов. Параметр Count содержит количество отображаемых элементов поля. Параметр Field содержит базовое имя поля данных
AutoSort		Order As Long, Field As String	Устанавливает автоматическую сортировку поля в зависимости от содержимого параметра Order (для установки параметра используются константы XlSortOrder) и базовых данных, на которые указывает параметр Field
Calculated-Items	Calculated-Items		Возвращает группу вычислимых элементов сводной таблицы, связанных с полем
Delete			Удаляет объект PivotField
PivotItems	Variant	[Index]	Возвращает объект или коллекцию, содержащие один (PivotItem) или группу (PivotItems) элементов сводной таблицы, связанных с полем

Пример: объект PivotField, коллекция PivotFields и коллекция CalculatedFields

```
Sub AddField()
   Dim oPT As PivotTable
   Dim oPF As PivotField
   Set oPT = ActiveSheet.PivotTables(1)
 ' Устанавливает аргумент UseStandardFormula в значение True
 ' При этом имена полей в формуле будут отформатированы
 ' в соответствии с региональными стандартами США, а не с
 ' текущими региональными параметрами. Обратите внимание,
```

```
  ' что запуск или отладка этой книги в версиях Excel до 2003
  ' приведет к сообщению об ошибке "Неправильное
  ' количество аргументов".
  Set oPF = oPT.CalculatedFields.Add("Total", "=Price * Volume", True)
  oPF.Orientation = xlDataField
End Sub
```

Объект PivotFormula и коллекция PivotFormulas

В коллекции PivotFormulas хранятся формулы, связанные со сводной таблицей. Каждый объект PivotFormula соответствует формуле, которая используется в отчете сводной таблицы. Родительским объектом коллекции PivotFormulas является объект PivotTable.

Кроме типичных атрибутов, коллекция PivotFormulas предоставляет метод Add и свойство Count. Свойство Count возвращает количество элементов в коллекции. Метод Add принимает в качестве параметра строку Formula и добавляет объект PivotFormula в коллекцию.

Общие свойства объекта PivotFormula

Определения свойств Application, Creator и Parent приводились в начале приложения.

Свойства объекта PivotFormula

Имя	*Тип возвращаемого значения*	*Описание*
Formula	String	Устанавливает/возвращает формулу, связанную с таблицей. В формуле должны использоваться ссылки в формате A1
Index	Long	Только чтение. Возвращает порядок обработки формулы в родительской коллекции
StandardFormula	String	Устанавливает/возвращает формулы с использованием региональных стандартов США
Value	String	Устанавливает/возвращает формулу, связанную с таблицей

Методы объекта PivotFormula

Имя	*Тип возвращаемого значения*	*Параметры*	*Описание*
Delete			Удаляет формулу из родительской коллекции

Объект PivotItem, коллекция PivotItems и коллекция CalculatedItems

В коллекции PivotItems хранятся записи данных из поля. В коллекции CalculatedItems хранятся вычислимые записи поля. Каждый объект PivotItem соответствует одной записи в поле данных. Родительским объектом коллекций PivotItems и CalculatedItems является объект PivotField.

Кроме типичных атрибутов, коллекции PivotItems и CalculatedItems предоставляют одно дополнительное свойство. Свойство Count возвращает количество объектов в коллекции. Кроме этого, метод Add коллекции PivotItems позволяет добавлять

новый элемент в коллекцию (при этом необходимо указать значение параметра Name). Метод Add коллекции CalculatedItems также позволяет добавлять элементы в коллекцию, но требует значений параметров Name и Formula.

Общие свойства объекта PivotItem

Определения свойств Application, Creator и Parent приводились в начале приложения.

Свойства объекта PivotItem

Имя	*Тип возвращаемого значения*	*Описание*
Caption	String	Устанавливает/возвращает текстовую метку, связанную с элементом
ChildItems	Variant	Только чтение. Параметры: [Index]. Возвращает объект или коллекцию, в которых содержится один (PivotItem) или группа (PivotItems) элементов сводной таблицы, связанных с этим элементом
DataRange	Range	Только чтение. Возвращает диапазон, в котором содержатся данные или элементы из этого элемента
DrilledDown	Boolean	Устанавливает/возвращает возможность разворачивания элемента сводной таблицы
Formula	String	Устанавливает/возвращает формулу, связанную с элементом
IsCalculated	Boolean	Только чтение. Возвращает информацию о том, является ли элемент вычисляемым полем данных
LabelRange	Range	Только чтение. Возвращает диапазон ячеек, в котором расположен элемент поля
Name	String	Устанавливает/возвращает имя элемента
ParentItem	PivotItem	Только чтение. Возвращает родительский элемент текущего элемента
ParentShow-Detail	Boolean	Только чтение. Возвращает информацию о том, отображается ли текущий элемент в результате отображения подробной структуры родительского объекта
Position	Long	Устанавливает/возвращает положение элемента среди других элементов с такой же ориентацией
RecordCount	Long	Только чтение. Возвращает количество записей в кэше сводной таблицы, содержащих данный элемент
ShowDetail	Boolean	Устанавливает/возвращает необходимость отображения подробной структуры элемента
SourceName	Variant	Только чтение. Возвращает имя исходных данных элемента
SourceName-Standard	String	Только чтение. Возвращает исходное имя элемента сводной таблицы, соответствующее региональным стандартам США
Standard-Formula	String	Устанавливает/возвращает формулы с использованием региональных стандартов США
Value	String	Устанавливает/возвращает имя указанного элемента
Visible	Boolean	Устанавливает/возвращает видимость элемента

Методы объекта PivotItem

Имя	Тип возвращаемого значения	Параметры	Описание
`Delete`			Удаляет элемент из коллекции

Пример: объект PivotItem, коллекция PivotItems и коллекция CalculatedItems

```
Sub ShowPivotItemData()
   Dim oPT As PivotTable
   Dim oPI As PivotItem
   ' Получить сводную таблицу
   Set oPT = ActiveSheet.PivotTables(1)
   ' Получить элемент сводной таблицы
   Set oPI = oPT.PivotFields("Product").PivotItems("Oranges")
   ' Отображает все строки исходных данных для этого элемента
   oPI.ShowDetail = True
End Sub
```

Объект PivotItemList

Данный объект представляет список элементов, связанных с определенной ячейкой сводной таблицы. Доступ к этому списку осуществляется через объект `PivotCell`. Объект `PivotItemList` доступен через свойство `ColumnItems` или `RowItems` объекта `PivotCell`. Количество элементов строк или столбцов в составе объекта `PivotItemList` зависит от структуры сводной таблицы.

Например, ячейка D5 в сводной таблице называется WroxSales1. В области строк слева от ячейки D5 находится заголовок строки OR (Oregon). Слева от заголовка OR находится еще одна метка строки, которая называется Region1. На основе этой информации код выдаст значение 2:

```
MsgBox wksPivotTable.Range("D5").PivotCell.RowItems.Count
```

Следующий код выдаст `Region1`. Это самая левая метка в строке ячейки D5.

```
MsgBox wksPivotTable.Range("D5").PivotCell.RowItems(1)
```

Наконец, следующий код выдаст `OR`. Это вторая метка слева в строке ячейки D5.

```
MsgBox wksPivotTable.Range("D5").PivotCell.RowItems(2)
```

Практическое применение объектов `PivotItemList` и `PivotCell` еще не придумано. Обычно нужна обратная функциональность. Необходимо получить информацию, соответствующую определенным заголовкам строки и столбца. Такая функциональность обеспечивается методом `GetPivotData` и функцией листа `GETPIVOTDATA`.

Общие свойства объекта PivotItemList

Определения свойств `Application`, `Creator` и `Parent` приводились в начале приложения.

Свойства объекта PivotItemList

Имя	Тип возвращаемого значения	Описание
`Count`	`Long`	Возвращает количество объектов в коллекции

Методы объекта PivotItemList

Имя	*Тип возвращаемого значения*	*Параметры*	*Описание*
Item	PivotItem	Index As Variant	Возвращает единственный объект PivotItem из списка PivotItemList

Объект PivotLayout

Этот объект описывает расположение полей сводной диаграммы в родительской диаграмме. Объекты Chart и ChartGroup могут выступать в роли родительских объектов PivotChart.

Общие свойства объекта PivotLayout

Определения свойств Application, Creator и Parent приводились в начале приложения.

Свойства объекта PivotLayout

Имя	*Тип возвращаемого значения*	*Описание*
ColumnFields		Только чтение. Параметры: [Index]. Возвращает объект или коллекцию, содержащие поле (PivotField) или поля (PivotFields) сводной таблицы, соответствующие столбцам сводной диаграммы
CubeFields	CubeFields	Только чтение. Возвращает коллекцию полей куба, связанных со сводной диаграммой
DataFields		Только чтение. Параметры: [Index]. Возвращает объект или коллекцию, содержащие поле (PivotField) или поля (PivotFields) сводной таблицы, соответствующие полям данных сводной диаграммы
HiddenFields		Только чтение. Параметры: [Index]. Возвращает объект или коллекцию, содержащие поле (PivotField) или поля (PivotFields) сводной таблицы, соответствующие скрытым полям сводной диаграммы
InnerDetail	String	Устанавливает/возвращает имя поля, которое будет отображаться при установке свойства ShowDetail в значение True
PageFields		Только чтение. Параметры: [Index]. Возвращает объект или коллекцию, содержащие поле (PivotField) или поля (PivotFields) сводной таблицы, соответствующие полям страницы сводной диаграммы
PivotCache	PivotCache	Только чтение. Возвращает кэш данных сводной диаграммы
PivotFields		Только чтение. Параметры: [Index]. Возвращает объект или коллекцию, содержащие поле (PivotField) или поля (PivotFields) сводной таблицы, соответствующие полям страницы сводной диаграммы
PivotTable	PivotTable	Только чтение. Возвращает сводную таблицу, связанную со сводной диаграммой

Имя	*Тип возвращаемого значения*	*Описание*
RowFields		Только чтение. Параметры: [Index]. Возвращает объект или коллекцию, содержащие поле (PivotField) или поля (PivotFields) сводной таблицы, соответствующие строкам сводной диаграммы
VisibleFields		Только чтение. Параметры: [Index]. Возвращает объект или коллекцию, содержащие поле (PivotField) или поля (PivotFields) сводной таблицы, соответствующие видимым полям сводной диаграммы

Методы объекта PivotLayout

Имя	*Тип возвращаемого значения*	*Параметры*	*Описание*
AddFields		[RowFields], [ColumnFields], [PageFields], [AppendField]	Добавляет поля строк, столбцов и страниц в отчет сводной диаграммы. Параметрам RowFields, ColumnFields и PageFields присваивается одно строковое имя поля или массив строковых имен полей. Для добавления поля к диаграмме параметр AppendField устанавливается в значение True. Для замещения поля на диаграмме параметр AppendField устанавливается в значение False

Пример: объект PivotLayout

```
Sub SetPivotLayout()     Dim oPL As PivotLayout
   ' Получить макет сводной диаграммы
   Set oPL = Charts(1).PivotLayout
   ' Показать продажи апельсинов по региону
   With oPL
       .AddFields RowFields:="Регион", PageFields:="Продукт"
       .PageFields("Продукт").CurrentPage = "Апельсины"
   End With
End Sub
```

Объект PivotTable и коллекция PivotTables

В коллекции PivotTables хранятся сводные таблицы из родительского листа. Каждый объект PivotTable в коллекции поддерживает создание и управление сводными таблицами Excel. Родительским объектом коллекции PivotTables является объект Worksheet.

Кроме типичных атрибутов, коллекция PivotTables предоставляет свойство Count и метод Add. Свойство Count возвращает количество объектов PivotTable в коллекции. Метод Add принимает в качестве параметров новый кэш сводной таблицы (содержащий данные) и диапазон ячейки назначения, который определяет верхний левый угол отчета сводной таблицы и создает новый отчет сводной таблицы. Кроме этого, метод Add поддерживает указание имени нового отчета сводной таблицы.

Общие свойства объекта PivotTable

Определения свойств Application, Creator и Parent приводились в начале приложения.

Свойства объекта PivotTable

Имя	*Тип возвращаемого значения*	*Описание*
CacheIndex	Long	Устанавливает/возвращает номер кэша текущей сводной таблицы
CalculatedMembers	CalculatedMembers	Только чтение. Возвращает все вычислимые поля и вычислимые элементы текущей сводной таблицы
ColumnFields	Object	Только чтение. Параметры: [Index]. Возвращает объект или коллекцию, содержащие поле (PivotField) или поля (PivotFields) сводной таблицы, соответствующие столбцам сводной таблицы
ColumnGrand	Boolean	Устанавливает/возвращает необходимость отображения итогов для столбцов сводной таблицы
ColumnRange	Range	Только чтение. Возвращает диапазон ячеек, содержащих область столбцов из отчета сводной таблицы
CubeFields	CubeFields	Только чтение. Возвращает коллекцию полей куба, связанных с отчетом сводной таблицы
DataBodyRange	Range	Только чтение. Возвращает диапазон ячеек, в котором расположена область данных отчета сводной таблицы
DataFields	Object	Только чтение. Параметры: [Index]. Возвращает объект или коллекцию, содержащие поле (PivotField) или поля (PivotFields) сводной таблицы, соответствующие полям данных сводной таблицы
DataLabelRange	Range	Только чтение. Возвращает диапазон ячеек, в котором хранятся метки полей данных отчета сводной таблицы
DataPivotField	PivotField	Только чтение. Возвращает все поля данных из сводной таблицы
DisplayEmptyColumn	Boolean	Только чтение. Возвращает наличие непустого ключевого слова многомерного выражения в запросе к поставщику OLAP для оси значений
DisplayEmptyRow	Boolean	Только чтение. Возвращает наличие непустого ключевого слова многомерного выражения в запросе к поставщику OLAP для оси категорий
DisplayErrorString	Boolean	Устанавливает/возвращает необходимость отображения строки из свойства ErrorString в ячейках, содержащих ошибки
DisplayImmediateItems	Boolean	Устанавливает/возвращает необходимость отображения областей строк и столбцов, когда область данных сводной таблицы пуста
DisplayNullString	Boolean	Устанавливает/возвращает необходимость вывода строки из свойства NullString в ячейках, содержащих значение null
EnableValueEditing	Boolean	Устанавливает/возвращает необходимость вывода предупреждения при перезаписи значений в области данных сводной таблицы

Имя	*Тип возвращаемого значения*	*Описание*
EnableDrilldown	Boolean	Устанавливает/возвращает возможность развертывания отчета сводной таблицы
EnableFieldDialog	Boolean	Устанавливает/возвращает необходимость отображения диалогового окна для поля сводной таблицы после двойного щелчка на поле сводной таблицы
EnableFieldList	Boolean	Устанавливает/возвращает необходимость отключения списка полей сводной таблицы. Если на момент изменения свойства список показан на экране, то список исчезает
EnableWizard	Boolean	Устанавливает/возвращает доступность мастера сводной таблицы
ErrorString	String	Устанавливает/возвращает строку, которая отображается в ячейках с ошибками. Используется вместе со свойством DisplayErrorString
GrandTotalName	String	Устанавливает/возвращает строку, которая отображается в заголовке столбца или строки итогов отчета сводной таблицы. По умолчанию принято значение "Grand Total"
HasAutoFormat	Boolean	Устанавливает/возвращает необходимость автоматического переформатирования отчета сводной таблицы при обновлении данных или перемещении полей
HiddenFields	Object	Только чтение. Параметры: [Index]. Возвращает объект или коллекцию, содержащие поле (PivotField) или поля (PivotFields) сводной таблицы, соответствующие скрытым полям сводной таблицы
InnerDetail	String	Устанавливает/возвращает имя поля, в котором будет показана дополнительная информация при установке свойства ShowDetail в значение True
MDX	String	Только чтение. Возвращает многомерное выражение, которое отправляется поставщику для наполнения текущего представления сводной таблицы
ManualUpdate	Boolean	Устанавливает/возвращает необходимость ручного пересчета отчета сводной таблицы
MergeLabels	Boolean	Устанавливает/возвращает необходимость использования объединенных ячеек в элементах внешней строки, элементах столбцов, итогах и метках общей суммы отчета сводной таблицы
Name	String	Устанавливает/возвращает имя отчета сводной таблицы
NullString	String	Устанавливает/возвращает строку, которая отображается в ячейке со значением null. Используется совместно со свойством DisplayNullString
PageFieldOrder	Long	Устанавливает/возвращает способ добавления новых полей страницы в макет отчета сводной таблицы. Для установки этого свойства используются константы XLOrder
PageFields	Object	Только чтение. Параметры: [Index]. Возвращает объект или коллекцию, содержащие поле (PivotField) или поля (PivotFields) сводной таблицы, соответствующие полям страницы сводной таблицы

Имя	*Тип возвращаемого значения*	*Описание*
PageField-Style	String	Устанавливает/возвращает стиль, который используется в области полей страницы сводной таблицы
PageField-WrapCount	Long	Устанавливает/возвращает количество полей страницы в каждом столбце или строке отчета сводной таблицы
PageRange	Range	Только чтение. Возвращает диапазон, в котором содержится область страницы из отчета сводной таблицы
PageRange-Cells	Range	Только чтение. Возвращает диапазон, в котором содержатся поля страниц и раскрывающиеся списки с элементами отчета сводной таблицы
Pivot-Formulas	PivotFormulas	Только чтение. Возвращает коллекцию формул, которые используются в отчете сводной таблицы
Pivot-Selection	String	Устанавливает/возвращает выделение данных и меток в сводной таблице с использованием стандартного формата выделения в отчете сводной таблицы. Например, для выделения данных и метки для страны (`Country`) "Canada" строка должна выглядеть как "Country [Canada]"
Pivot-Selection-Standard	String	Устанавливает/возвращает выделение сводной таблицы в стандартном формате отчета сводной таблицы с использованием региональных стандартов США
Preserve-Formatting	Boolean	Устанавливает/возвращает необходимость сохранения форматирования отчета сводной таблицы при модификации, сортировке, повороте, обновлении или пересчете
PrintTitles	Boolean	Устанавливает/возвращает необходимость печати заголовка отчета сводной таблицы при печати родительского листа
RefreshDate	Date	Только чтение. Возвращает дату последнего обновления данных в отчете сводной таблицы
RefreshName	String	Только чтение. Возвращает имя пользователя, который последний раз обновлял данные в отчете сводной таблицы
RepeatItems-OnEach-PrintedPage	Boolean	Устанавливает/возвращает необходимость отображения меток строк, столбцов и элементов в первой строке каждой страницы при печати отчета сводной таблицы
RowFields		Только чтение. Параметры: `[Index]`. Возвращает объект или коллекцию, содержащие поле (`PivotField`) или поля (`PivotFields`) сводной таблицы, соответствующие строкам сводной таблицы
RowGrand	Boolean	Устанавливает/возвращает необходимость отображения общей суммы для строк сводной таблицы
RowRange	Range	Только чтение. Возвращает диапазон ячеек, в котором расположена область строк отчета сводной таблицы
SaveData	Boolean	Устанавливает/возвращает необходимость сохранения данных отчета сводной таблицы вместе с книгой
Selection-Mode	XlPTSelection-Mode	Устанавливает/возвращает способ установки выделения в отчете сводной таблицы (например, `xlLabelOnly`)
ShowCell-Backgroud-FromOLAP	Boolean	Устанавливает/возвращает необходимость включения в запрашиваемое Excel многомерное выражение свойства `BackColor` для каждой ячейки области данных, соответствующей ячейке в наборе данных OLAP

Имя	*Тип возвращаемого значения*	*Описание*
SmallGrid	Boolean	Устанавливает/возвращает необходимость использования таблицы 2x2 для созданного отчета сводной таблицы (True). В противном случае будет использоваться пустой контур (False)
SourceData	Variant	Устанавливает/возвращает источник данных отчета сводной таблицы. Это может быть ссылка на ячейку, массив, несколько ячеек или другой отчет сводной таблицы. Не используется с источниками данных OLE DB
Subtotal-HiddenPage-Items	Boolean	Устанавливает/возвращает необходимость включения скрытых полей страницы в итоги строки и столбца, итоги блоков и общие суммы
TableRange1	Range	Только чтение. Возвращает диапазон, содержащий весь отчет сводной таблицы, не включая поля страницы
TableRange2	Range	Только чтение. Возвращает диапазон, содержащий весь отчет сводной таблицы вместе с полями страницы
TableStyle	String	Устанавливает/возвращает стиль тела таблицы отчета
Tag	String	Устанавливает/возвращает строку, которая будет сохранена вместе с отчетом сводной таблицы (например, описание отчета сводной таблицы)
VacatedStyle	String	Устанавливает/возвращает используемый стиль аннулированных ячеек при обновлении отчета сводной таблицы
Value	String	Устанавливает/возвращает имя отчета сводной таблицы
Version	XlPivotTable-VersionList	Только чтение. Возвращает номер версии Excel
View-Calculated-Members	Boolean	Устанавливает/возвращает возможность просмотра вычислимых полей сводных таблиц OLAP
Visible-Fields		Только чтение. Параметры: [Index]. Возвращает объект или коллекцию, содержащие поле (PivotField) или поля (PivotFields) сводной таблицы, соответствующие видимым полям сводной таблицы
VisualTotals	Boolean	Устанавливает/возвращает необходимость пересчета сумм при сокрытии элемента сводной таблицы

Методы объекта PivotTable

Имя	*Тип возвращаемого значения*	*Параметры*	*Описание*
AddDataField	PivotField	Field As Object, [Caption], [Function]	Добавляет поле данных в отчет сводной таблицы. Параметр Field содержит уникальное поле на сервере. Параметр Caption содержит метку, используемую для идентификации поля данных. Параметр Function содержит функцию, которая выполняется в добавленном поле данных

Имя	*Тип возвращаемого значения*	*Параметры*	*Описание*
AddFields	Variant	[RowFields], [ColumnFields], [PageFields], [AddToTable]	Добавляет поле строки, столбца и страницы в отчет сводной таблицы. Параметры RowFields, ColumnFields и PageFields могут содержать одну строку с именем поля или массив строк с именами полей. Для добавления полей в отчет параметр AddToTable устанавливается в значение True. Для замещения полей в отчете параметр AddToTable устанавливается в значение False
CalculatedFields	CalculatedFields		Возвращает коллекцию вычислимых полей отчета сводной таблицы
CreateCubeFile	String	File As String, [Measures], [Levels], [Members], [Properties]	Создает файл куба на основе отчета сводной таблицы, связанного с источником данных OLAP. Параметр File содержит имя создаваемого файла куба. Параметр Measures содержит массив уникальных имен измерений, которые будут частью среза. Параметр Levels содержит массив строк, в котором каждый элемент является уникальным именем уровня. Параметр Members содержит массив строковых массивов, в котором каждый элемент соответствует иерархиям, описанным в массиве из параметра Levels. Если свойства Members не должны быть включены в срез, параметр Properties устанавливается в значение False
Format		Format As xlPivotFormatType	Выбирает предопределенный стиль форматирования отчета сводной таблицы, указанный в параметре Format
GetData	Double	Name As String	Устанавливает значение конкретной ячейки отчета сводной таблицы. Значение параметра Name должно соответствовать стандартному формату выделения отчета сводной таблицы

Имя	*Тип возвращаемого значения*	*Параметры*	*Описание*
GetPivotData	Range	[DataField], [Field1], [Item1], [Field2], [Item2], [Field3], [Item3], [Field4], [Item4], [Field5], [Item5], [Field6], [Item6], [Field7], [Item7], [Field8], [Item8], [Field9], [Item9], [Field10], [Item10], [Field11], [Item11], [Field12], [Item12], [Field13], [Item13], [Field14], [Item14]	Возвращает информацию об элементе данных отчета сводной таблицы. Параметр FieldN содержит имя поля столбца или строки в отчете сводной таблицы, а параметр ItemN содержит имя элемента в поле Field14
ListFormulas			Создает отдельный лист со списком всех вычислимых элементов и полей сводной таблицы
PivotCache	PivotCache		Возвращает кэш данных, связанных с текущей сводной таблицей
PivotFields	Object	[Index]	Возвращает объект или коллекцию, содержащие поле (PivotField) или поля (PivotFields) сводной таблицы, соответствующие полям сводной таблицы
PivotSelect		Name As String, [Mode As XlPTSelectionMode], [UseStandardName]	Выделяет фрагмент сводной таблицы, на который указывает параметр Name. Этот параметр должен соответствовать стандартному формату выделения отчета сводной таблицы. Параметр Mode определяет выделяемую часть сводной таблицы (например, xlBlanks). Для записи макросов, которые будут работать с другими региональными параметрами, параметр UseStandardName устанавливается в значение True

Имя	*Тип возвращаемого значения*	*Параметры*	*Описание*
PivotTable-Wizard		[SourceType], [SourceData], [Table-Destination], [TableName], [RowGrand], [ColumnGrand], [SaveData], [HasAutoFormat], [AutoPage], [Reserved], [Background-Query], [OptimizeCache], [PageField-Order], [PageFieldWrap-Count], [ReadData], [Connection]	Создает отчет сводной таблицы. С помощью констант XLPivotTa-bleSourceType параметр Source-Type определяет тип источника данных, который используется для построения сводной диаграммы. В параметре TableDestination передается диапазон родительского листа, в котором будет размещен создаваемый отчет. В параметре TableName содержится имя создаваемого отчета. Для отображения общей суммы для строк или столбцов в значение True устанавливаются параметр RowGrand или Col-umnGrand. Для автоматического форматирования отчета при обновлении или изменении параметр HasAutoFormat устанавливается в значение True. Параметр Auto-Page позволяет включить автоматическое создание поля страницы для консолидации. Для асинхронной работы с источником данных свойство BackgroundQuery устанавливается в значение True. Для оптимизации кэша при построении параметр Op-timizeCache устанавливается в значение True. Присвоение констант XLOrder параметру Page-FieldOrder позволяет настроить способ добавления полей в отчет. Параметр PageFieldWrapCount позволяет установить количество полей страницы в каждом столбце или в каждой строке. Установка параметра ReadData в значение True позволяет копировать данные из внешней базы данных в кэш. Наконец, параметр Connection позволяет указать строку подключения ODBC для кэша сводной таблицы
RefreshTable	Boolean		Обновляет отчет сводной таблицы на основе исходных данных. При успешном обновлении возвращает значение True
ShowPages	Variant	[PageField]	Создает новый отчет сводной таблицы на новом листе для каждого элемента в поле страницы (PageField)
Update			Обновляет отчет сводной таблицы

Пример: объект PivotTable и коллекция PivotTables

```
Sub PreviewPivotTable()
    Dim oPT As PivotTable
   ' Получить макет сводной таблицы
   Set oPT = ActiveSheet.PivotTables(1)
   ' Добавить заголовки строк и столбцов. После этого
   ' включить предварительный просмотр таблицы
   With oPT
      .ColumnGrand = False
      .RowGrand = True
      .TableRange2.PrintPreview
   End With
End Sub
```

Объект PlotArea

В объекте `PlotArea` хранятся параметры форматирования, связанные с областью построения родительской диаграммы. Для двумерных диаграмм в объекте `PlotArea` хранятся линии тренда, маркеры данных, направляющие, метки данных и метки осей (но не заголовки). Для трехмерных диаграмм в объекте `PlotArea` хранятся вертикальные плоскости, базовые уровни, заголовки осей, маркеры и все элементы, хранящиеся для двумерных диаграмм. Вокруг области построения расположена область диаграммы. Дополнительная информация о параметрах форматирования области диаграммы рассматривается в разделе, посвященном объекту `ChartArea`. Объект `Chart` всегда является родительским объектом объекта `PlotArea`.

Общие свойства объекта PlotArea

Определения свойств `Application`, `Creator` и `Parent` приводились в начале приложения.

Свойства объекта PlotArea

Имя	*Тип возвращаемого значения*	*Описание*
`Border`	`Border`	Только чтение. Возвращает свойства границы вокруг области построения
`Fill`	`ChartFillFormat`	Только чтение. Возвращает объект с параметрами форматирования заливки области построения
`Height`	`Double`	Устанавливает/возвращает высоту области построения диаграммы
`InsideHeight`	`Double`	Только чтение. Возвращает высоту внутренней части области построения без меток осей
`InsideLeft`	`Double`	Только чтение. Возвращает расстояние от левого края области построения, не включая метки осей, до левого края диаграммы
`InsideTop`	`Double`	Только чтение. Возвращает расстояние от верхнего края области построения, не включая метки осей, до верхнего края диаграммы
`InsideWidth`	`Double`	Только чтение. Возвращает ширину внутренней части области построения без меток осей

Имя	*Тип возвращаемого значения*	*Описание*
Interior	Interior	Только чтение. Возвращает объект, содержащий параметры форматирования внутренней части области построения (например, цвет внутренней области)
Left	Double	Устанавливает/возвращает расстояние от левого края области построения до левого края диаграммы
Name	String	Только чтение. Возвращает имя области построения
Top	Double	Устанавливает/возвращает расстояние от верхнего края области построения до верхнего края диаграммы
Width	Double	Устанавливает/возвращает ширину области построения

Методы объекта PlotArea

Имя	*Тип возвращаемого значения*	*Параметры*	*Описание*
ClearFormats	Variant		Очищает форматирование области построения
Select	Variant		Выделяет область построения диаграммы

Пример: объект PlotArea

В этом примере объект PlotArea используется для настройки одинакового размера и положения области построения для всех диаграмм в пределах книги в независимости от форматирования осей (например, оси могут использовать разные шрифты или единицы измерения).

```
Sub MakeChartAreasSameSizeAsFirst()
    Dim oCht As Chart, oPA As PlotArea
   Dim dWidth As Double, dHeight As Double
   Dim dTop As Double, dLeft As Double
   ' Получить размеры внутренней части области построения
   ' первой диаграммы
   With Charts(1).PlotArea
      dWidth = .InsideWidth
      dHeight = .InsideHeight
      dLeft = .InsideLeft
      dTop = .InsideTop
   End With
   ' Перебрать все диаграммы в книге
   For Each oCht In Charts
      ' Получить объект PlotArea
      Set oPA = oCht.PlotArea

      ' Изменить размер и положение области построения
      With oPA
         If .InsideWidth > dWidth Then
            ' Слишком большая. Сделать меньше.
            .Width = .Width - (.InsideWidth - dWidth)
         Else
            ' Слишком маленькая. Переместить левее и сделать больше
            .Left = .Left - (dWidth - .InsideWidth)
            .Width = .Width + (dWidth - .InsideWidth)
         End If

         If .InsideHeight > dHeight Then
            ' Слишком большая. Сделать меньше
```

```
            .Height = .Height - (.InsideHeight - dHeight)
        Else
            ' Слишком маленькая. Переместить выше и сделать больше.
            .Top = .Top - (dHeight - .InsideHeight)
            .Height = .Height + (dHeight - .InsideHeight)
        End If
        ' Установить положение внутренней части области построения
        .Left = .Left + (dLeft - .InsideLeft)
        .Top = .Top + (dTop - .InsideTop)
    End With
  Next
End Sub
```

Объект Point и коллекция Points

В коллекции Points хранятся все точки данных определенного ряда на диаграмме. На самом деле диаграмма (объект Chart) может содержать несколько групп диаграммы (ChartGroups/ChartGroup), в которых хранятся несколько рядов (SeriesCollection/Series). В свою очередь, ряды содержат множество точек (Points/Point). Объект Point описывает определенную точку ряда. Родительским для коллекции Points является объект Series.

Кроме типичных атрибутов, коллекция Points предоставляет свойство Count. Это свойство возвращает количество объектов Point в коллекции.

Общие свойства объекта Point

Определения свойств Application, Creator и Parent приводились в начале приложения.

Свойства объекта Point

Имя	*Тип возвращаемого значения*	*Описание*
ApplyPicToEnd	Boolean	Устанавливает/возвращает необходимость добавления рисунка в конец точки
ApplyPicToFront	Boolean	Устанавливает/возвращает необходимость добавления рисунка в начало точки
ApplyPicToSides	Boolean	Устанавливает/возвращает необходимость добавления рисунка по сторонам точки
Border	Border	Только чтение. Возвращает свойства границы вокруг точки
DataLabel	DataLabel	Только чтение. Возвращает объект, поддерживающий управление атрибутами метки данных (например, форматированием, текстом). Используется совместно со свойством HasDataLabel
Explosion	Long	Устанавливает/возвращает расстояние, на которое выдвигается доля (точка) круговой или кольцевой диаграммы. Для отключения выдвижения свойство устанавливается в значение 0
Fill	ChartFillFormat	Только чтение. Возвращает объект, содержащий параметры форматирования заливки для точки
HasDataLabel	Boolean	Устанавливает/возвращает необходимость отображения метки данных вместе с точкой. Используется совместно со свойством DataLabel

Имя	*Тип возвращаемого значения*	*Описание*
Interior	Interior	Только чтение. Возвращает объект, содержащий параметры форматирования внутренней области точки (например, цвет внутренней области)
InvertIf-Negative	Boolean	Устанавливает/возвращает необходимость инвертирования цвета при отрицательном значении точки
Marker-Background-Color	Long	Устанавливает/возвращает цвет фона маркера точки. Для создания значения цвета используется функция RGB
Marker-Background-ColorIndex	XlColorIndex	Устанавливает/возвращает цвет фона маркера точки. Для установки этого свойства используются константы XlColorIndex или значение индекса цвета в текущей палитре
MarkerSize	Long	Устанавливает/возвращает размер маркера ключа точки
MarkerStyle	XlMarkerStyle	Устанавливает/возвращает тип маркера, который используется в качестве ключа (например, квадрат, ромб, треугольник, рисунок и т.д.)
PictureType	XlChart-PictureType	Устанавливает/возвращает способ отображения рисунка возле точки (например, с растягиванием, в виде мозаики). Для установки этого свойства используются константы XlPictureType
PictureUnit	Long	Устанавливает/возвращает количество единиц, которые представляет рисунок, если свойство PictureType установлено в значение xlScale
Secondary-Plot	Boolean	Устанавливает/возвращает расположение точки на второй части диаграммы "круговая в круговой" или "гистограмма в круговой"
Shadow	Boolean	Устанавливает/возвращает необходимость отображения эффекта тени вокруг точки

Методы объекта Point

Имя	*Тип возвращаемого значения*	*Параметры*	*Описание*
ApplyData-Labels	Variant	[Type As XlDataLabelsType], [LegendKey], [AutoText], [HasLeaderLines], [ShowSeriesName], [ShowCategoryName], [ShowValue], [ShowPercentage], [ShowBubbleSize], [Separator]	Применяет к точке свойства метки данных, описанные параметрами метода. Параметр Type определяет отсутствие метки, вывод значения, вывод доли целого в процентах или вывод метки категории. Возле точки может отображаться ключ легенды. Для этого параметр LegendKey необходимо установить в значение True. Если объект автоматически генерирует соответствующий текст на основе содержимого, параметр AutoText устанавливается в значение True. Если ряд содержит линии меток, параметр HasLeaderLines должен быть установлен в значение True. Все остальные параметры являются свойствами описываемой метки данных

Имя	*Тип возвращаемого значения*	*Параметры*	*Описание*
`ClearFormats`	`Variant`		Очищает форматирование точки
`Copy`	`Variant`		Копирует точку в буфер обмена
`Delete`	`Variant`		Удаляет точку
`Paste`	`Variant`		Вставляет рисунок из буфера обмена в текущую точку, превращая рисунок в маркер
`Select`	`Variant`		Выделяет точку на диаграмме

Пример: объект Point и коллекция Points

```
Sub ExplodePie()
   Dim oPt As Point
   ' Получить первую точку данных на круговой диаграмме
   Set oPt = Charts(1).SeriesCollection(1).Points(1)
   ' Добавить метку к первой точке и отодвинуть ее от диаграммы
   With oPt
      .ApplyDataLabels xlDataLabelsShowLabelAndPercent
      .Explosion = 20
   End With
End Sub
```

Объект Protection

Этот объект предоставляет доступ к группе параметров защиты листа, появившихся в Excel 2003. При защите листа можно указать такие возможности: выделение незаблокированных ячеек, форматирование ячеек, столбцов и строк, вставка и удаление строк и столбцов, сортировка и т.д.

Установка параметров защиты выполняется с помощью метода `Protect` объекта `Worksheet`. Для проверки текущих параметров защиты можно воспользоваться свойством `Protection` объекта `Worksheet`.

```
MsgBox ActiveSheet.Protection.AllowFormattingCells
```

Свойства объекта Protection

Имя	*Тип возвращаемого значения*	*Описание*
`AllowDeletingColumns`	`Boolean`	Только чтение. Возвращает возможность удаления столбцов на защищенном листе
`AllowDeletingRows`	`Boolean`	Только чтение. Возвращает возможность удаления строк на защищенном листе
`AllowEditRanges`	`AllowEditRanges`	Только чтение. Возвращает объект `AllowEditRanges`
`AllowFiltering`	`Boolean`	Только чтение. Возвращает возможность использования автофильтра, который создавался до установки защиты на лист
`AllowFormattingCells`	`Boolean`	Только чтение. Возвращает возможность форматирования ячеек на защищенном листе

Имя	Тип возвращаемого значения	Описание
AllowFormatingColumns	Boolean	Только чтение. Возвращает возможность форматирования столбцов на защищенном листе
AllowFormatingRows	Boolean	Только чтение. Возвращает возможность форматирования строк на защищенном листе
AllowInsertingColumns	Boolean	Только чтение. Возвращает возможность вставки столбцов в защищенный лист
AllowInsertingHyperlinks	Boolean	Только чтение. Возвращает возможность вставки гиперссылок в защищенный лист
AllowInsertingRows	Boolean	Только чтение. Возвращает возможность вставки строк в защищенный лист
AllowSorting	Boolean	Только чтение. Возвращает возможность сортировки защищенного листа
AllowUsingPivotTables	Boolean	Только чтение. Возвращает возможность управления сводными таблицами на защищенном листе

Пример: объект Protection

В следующей подпрограмме параметры защиты устанавливаются на основе имени пользователя с вкладки Общие (General) в диалоговом окне Сервис⇨Параметры (Tools⇨Options), а также на основе параметров пользователя, указанных в таблице на листе. Если пользователь не найден, об этом выдается сообщение и применяются принятые по умолчанию параметры.

```
Sub ProtectionSettings()
    Dim rngUsers As Range, rngUser As Range
    Dim sCurrentUser As String

    ' Получить текущее имя пользователя
    sCurrentUser = Application.UserName

    ' Определить список пользователей в таблице
    With wksAllowEditRange
        Set rngUsers = .Range(.Range("Users"),
↳.Range("Users").End(xlToRight))
    End With

    ' Найти текущего пользователя в таблице
    Application.FindFormat.Clear
    Set rngUser = rngUsers.Find(What:=sCurrentUser, SearchOrder:=xlByRows,
↳MatchCase:=False, SearchFormat:=False)

    ' Присутствует ли пользователь в таблице?
    If Not rngUser Is Nothing Then
        ' Установить свойства объекта Protection
        ' в сответствии с параметрами в таблице
        wksAllowEditRange.Protect Password:="wrox1", _
        DrawingObjects:=True, _
        Contents:=True, _
        AllowFormattingCells:=rngUser.Offset(1, 0).Value, _
        AllowFormattingColumns:=rngUser.Offset(2, 0).Value, _
```

```
        AllowFormattingRows:=rngUser.Offset(3, 0).Value, _
        AllowSorting:=rngUser.Offset(4, 0).Value, _
        UserInterfaceOnly:=True

        ' Выделение незаблокированных ячеек, заблокированных
        ' и незаблокированных и выключение обеих возможностей
        ' не являются частью объекта Protection
        If rngUser.Offset(5, 0).Value = True Then
            wksAllowEditRange.EnableSelection = xlUnlockedCells
        Else
            wksAllowEditRange.EnableSelection = xlNoRestrictions
        End If
    Else
        ' Текущий пользователь отсутствует в таблице
        MsgBox "Пользователь не найден в таблице. Используются
➥ принятые по умолчанию параметры.", vbExclamation, "Protection
➥ Settings"
        wksAllowEditRange.Protect , True, True, False, False, False, _
                                  False, False, False, False, False, _
                                  False, False, False, False, False

        wksAllowEditRange.EnableSelection = xlNoRestrictions

    End If

End Sub
```

Объект PublishObject и коллекция PublishObjects

В коллекции `PublishObjects` хранятся все сохраненные на Web-странице элементы книги. В каждом объекте `PublishObject` хранится элемент книги, который был сохранен на Web-странице и может потребоваться для обновления значений на Web-странице. Родительским объектом коллекции `PublishObjects` является объект `Workbook`.

Кроме типичных атрибутов, коллекция `PublishObjects` предоставляет несколько свойств и методов. Уникальные атрибуты перечислены в следующих таблицах.

Свойства и методы коллекции PublishObjects

Имя	*Тип возвращаемого значения*	*Описание*
`AutoRepublish`	`Boolean`	Устанавливает/возвращает необходимость повторной публикации элементов коллекции `PublishObjects` при сохранении книги
`Count`	`Long`	Только чтение. Возвращает количество объектов `PublishObject` в коллекции
`Add`	`PublishObject`	Метод. Параметры: `SourceType As XlSourceType`, `Filename As String`, `[Sheet]`, `[Source]`, `[HtmlType]`, `[DivID]`, `[Title]`. Добавляет в коллекцию объект `PublishObject`
`Delete`		Метод. Удаляет объект `PublishObject` из коллекции
`Publish`		Метод. Публикует на Web-странице все элементы, с которыми связаны объекты `PublishObject`

Общие свойства объекта PublishObject

Определения свойств `Application`, `Creator` и `Parent` приводились в начале приложения.

Свойства объекта PublishObject

Имя	*Тип возвращаемого значения*	*Описание*
`DivID`	`String`	Только чтение. Возвращает идентификатор, который используется в теге `<DIV>` в коде Web-страницы
`Filename`	`String`	Устанавливает/возвращает адрес URL или путь, по которому объект будет сохранен в виде Web-страницы
`HtmlType`	`XlHtmlType`	Устанавливает/возвращает тип сохраняемой Web-страницы (например, `xlHtmlStatic`, `xlHtmlChart`). Все страницы, кроме типа `xlHtmlStatic`, нуждаются в особых компонентах ActiveX
`Sheet`	`String`	Только чтение. Возвращает лист Excel, который сохраняется в виде Web-страницы
`Source`	`String`	Только чтение. возвращает конкретный элемент, например имя диапазона, имя диаграммы или имя отчета из базового типа, указанного в свойстве `SourceType`
`SourceType`	`XlSourceType`	Только чтение. Возвращает тип публикуемого источника (например, `xlSourceChart`, `xlSourcePrintArea` и т.д.)
`Title`	`String`	Устанавливает/возвращает заголовок публикуемой Web-страницы

Методы объекта PublishObject

Имя	*Тип возвращаемого значения*	*Параметры*	*Описание*
`Delete`			Удаляет объект `PublishObject`
`Publish`		`[Create]`	Публикует исходные элементы объекта `PublishObject` на Web-странице. Для перезаписи существующих файлов параметр `Create` устанавливается в значение `True`. При установке параметра в значение `False` публикуемые элементы будут добавлены к одноименной Web-странице

Пример: объект PublishObject и коллекция PublishObjects

```
Sub UpdatePublishedCharts()
   Dim oPO As PublishObject
   For Each oPO In ActiveWorkbook.PublishObjects
      If oPO.SourceType = xlSourceChart Then
         oPO.Publish
      End If
   Next
End Sub
```

Объект QueryTable и коллекция QueryTables

В коллекции QueryTables хранятся таблицы данных, созданные на основе внешних источников данных. Каждому объекту QueryTable соответствует одна таблица на листе, содержащая данные из внешнего источника, в качестве которого могут выступать источники ODBC, OLE DB, текстовые файлы, Data Finder, Web-запрос или набор записей DAO/ADO. Родительским для коллекции QueryTables является объект Worksheet.

Кроме типичных атрибутов, коллекция QueryTables предоставляет несколько свойств и методов. Уникальные атрибуты показаны ниже.

Свойства и методы коллекции QueryTables

Имя	*Тип возвращаемого значения*	*Описание*
Count	Long	Только чтение. Возвращает количество элементов коллекции
Add	QueryTable	Метод. Параметры: Connection, Destination As Range, [Sql]. Добавляет в коллекцию объект QueryTable. В качестве значения параметра Connection можно указать строку подключения ODBC или OLE DB, другой объект QueryTable, объект набора записей DAO или ADO, Web-запрос, строку Data Finder или имя текстового файла. Значение параметра Destination определяет положение верхнего левого угла результатов таблицы запроса. Параметр SQL может содержать запрос SQL для подключения

Общие свойства объекта QueryTable

Определения свойств Application, Creator и Parent приводились в начале приложения.

Свойства объекта QueryTable

Имя	*Тип возвращаемого значения*	*Описание*
Adjust-ColumnWidth	Boolean	Устанавливает/возвращает необходимость автоматического изменения ширины столбцов в соответствии с объемом данных при каждом обновлении таблицы
Background-Query	Boolean	Устанавливает/возвращает необходимость асинхронной обработки таблицы запросов
CommandText	Variant	Устанавливает/возвращает запрос SQL, который используется для получения данных
CommandType	XlCmdType	Устанавливает/возвращает тип значения свойства CommandText (например, xlCmdSQL, xlCmdTable)
Connection	Variant	Устанавливает/возвращает строку подключения OLE DB, ODBC, источник данных в Web, путь к текстовому файлу или к базе данных
Destination	Range	Только чтение. Возвращает ячейку, в которой будет расположен верхний левый угол результатов таблицы запроса
EditWebPage	Variant	Устанавливает/возвращает адрес URL для Web-запроса
Enable-Editing	Boolean	Устанавливает/возвращает возможность редактирования таблицы запроса. При установке свойства в значение False таблица может только обновляться

Имя	*Тип возвращаемого значения*	*Описание*
Enable-Refresh	Boolean	Устанавливает/возвращает возможность обновления данных в таблице запроса
FetchedRow-Overflow	Boolean	Только чтение. Возвращает информацию о том, было ли в результате последнего обновления таблицы запроса получено больше строк, чем доступно на листе
FieldNames	Boolean	Устанавливает/возвращает необходимость использования названий полей из источника данных в виде заголовков столбцов в таблице запроса
FillAdjac-entFormulas	Boolean	Устанавливает/возвращает необходимость автоматического обновления расположенных справа от таблицы запроса формул при обновлении данных в таблице запроса
ListObjet	Range	Возвращает объект Range
Maintain-Connection	Boolean	Устанавливает/возвращает необходимость сохранения открытым подключения к источнику данных до закрытия листа. Используется только с источниками данных OLE DB
Name	String	Устанавливает/возвращает имя таблицы запроса
Parameters	Parameters	Только чтение. Возвращает параметры таблицы запроса
PostText	String	Устанавливает/возвращает текст запроса POST, который отправляется на Web-сервер для получения данных в результате Web-запроса
Preserve-ColumnInfo	Boolean	Устанавливает/возвращает необходимость сохранения расположения, порядка сортировки и фильтрации при обновлении запроса данных
Preserve-Formatting	Boolean	Устанавливает/возвращает необходимость применения к новым строкам таблицы запроса форматирования, связанного с первыми пятью строками данных
QueryType	xlQueryType	Только чтение. Возвращает тип подключения, связанного с таблицей запроса (например, xlOLEDBQuery, xlDAO-Query, xlTextImport)
Recordset		Только чтение. Возвращает набор записей, связанный с запросом к источнику данных
Refreshing	Boolean	Только чтение. Возвращает состояние асинхронного запроса
RefreshOn-FileOpen	Boolean	Устанавливает/возвращает необходимость обновления таблицы запроса при открытии книги
Refresh-Period	Long	Устанавливает/возвращает длительность (в минутах) между автоматическим обновлением данных из источника. Для отключения обновления это свойство устанавливается в значение 0
RefreshStyle	XlCell-InsertionMode	Устанавливает/возвращает способ обработки строк при извлечении данных из источника. Ячейки листа могут быть перезаписаны (xlOverwriteCells), строки могут частично вставляться/удаляться по необходимости (xlInsertDeleteCells) или добавляться (xlInsertEntireRows)
ResultRange	Range	Только чтение. Возвращает диапазон ячеек, содержащий результат таблицы запроса

Имя	*Тип возвращаемого значения*	*Описание*
RobustConnect	XlRobustConnect	Устанавливает/возвращает способ подключения кэша сводной таблицы к источнику данных
RowNumbers	Boolean	Устанавливает/возвращает необходимость добавления столбца с номерами строк слева от таблицы запроса
SaveData	Boolean	Устанавливает/возвращает необходимость сохранения данных таблицы запроса вместе с книгой
SavePassword	Boolean	Устанавливает/возвращает необходимость сохранения пароля подключения ODBC вместе с таблицей запроса
SourceConnectionFile	String	Устанавливает/возвращает имя файла, который использовался для создания сводной таблицы
SourceDataFile	String	Только чтение. Возвращает имя файла с исходными данными для кэша сводной таблицы
TextFileColumnDataTypes	Variant	Устанавливает/возвращает массив строковых констант, описывающих типы данных каждого столбца. Для этого применяются константы XlColumnDataType. Используется, когда свойство QueryType установлено в значение xlTextImport
TextFileCommaDelimiter	Boolean	Устанавливает/возвращает использование запятой в качестве разделителя данных. Применяется, если свойство QueryType установлено в значение xlTextImport и для текстовых файлов с разделителями
TextFileConsecutiveDelimiter	Boolean	Устанавливает/возвращает необходимость интерпретации идущих подряд разделителей (например, ",,,") в качестве одного разделителя. Используется, если свойство QueryType установлено в значение xlTextImport
TextFileDecimalSeparator	String	Устанавливает/возвращает тип используемого разделителя десятичной дроби. Применяется, если свойство QueryType установлено в значение xlTextImport
TextFileFixedColumnWidth	Variant	Устанавливает/возвращает массив значений ширины, соответствующих столбцам. Используется, если свойство QueryType установлено в значение xlTextImport и для текстовых файлов с фиксированным размером полей
TextFileOtherDelimiter	String	Устанавливает/возвращает символ, который будет использоваться в качестве разделителя столбцов в текстовом файле. Применяется, если свойство QueryType установлено в значение xlTextImport и для текстовых файлов с разделителями
TextFileParseType	XlTextParsingType	Устанавливает/возвращает тип импортируемого текстового файла: xlDelimited или xlFixedWidth. Используется, если свойство QueryType установлено в значение xlTextImport
TextFilePlatform	XlPlatform	Устанавливает/возвращает кодовые страницы, используемые при импорте текстового файла (например, xlMSDOS, xlWindows). Применяется, если свойство QueryType установлено в значение xlTextImport
TextFilePromptOnRefresh	Boolean	Устанавливает/возвращает необходимость запроса у пользователя текстового файла, импортируемого в таблицу запроса при каждом обновлении данных. Применяется, если свойство QueryType установлено в значение xlTextImport. Запрос не выдается при первом обновлении данных

Имя	*Тип возвращаемого значения*	*Описание*
TextFile-Semicolon-Delimiter	Boolean	Устанавливает/возвращает необходимость использования точки с запятой в качестве разделителя при импорте текстового файла. Применяется, если свойство QueryType установлено в значение xlTextImport
TextFile-Space-Delimiter	Boolean	Устанавливает/возвращает необходимость использования пробела в качестве разделителя при импорте текстового файла. Применяется, если свойство QueryType установлено в значение xlTextImport и файл является текстовым файлом с разделителями
TextFile-StartRow	Long	Устанавливает/возвращает номер строки, с которой начинается импорт текстового файла. Используется, если свойство QueryType установлено в значение xlTextImport
TextFile-Tab-Delimiter	Boolean	Устанавливает/возвращает необходимость использования символа табуляции при импорте текстового файла. Применяется, если свойство QueryType установлено в значение xlTextImport и файл является текстовым файлом с разделителями
TextFile-Text-Qualifier	XlText-Qualifier	Устанавливает/возвращает символ, используемый для определения данных строки при импорте данных из текстового файла. Применяется, если свойство QueryType установлено в значение xlTextImport
TextFile-Thousands-Separator	String	Устанавливает/возвращает символ, используемый в качестве разделителя тысяч при импорте из текстового файла (например, ‘,’)
TextFile-Trailing-MinusNumbers	Boolean	Устанавливает/возвращает необходимость интерпретации чисел, импортированных в виде текста, который начинается со знака ‘-‘, в виде отрицательных чисел
TextFile-VisualLayout	Long	Возвращает 1 или 2 в зависимости от визуального представления файла. Значение 1 соответствует направлению слева направо, а значение 2 — направлению справа налево
Web-Consecutive-Delimiters-AsOne	Boolean	Устанавливает/возвращает необходимость интерпретации нескольких разделителей подряд как одного разделителя при импорте данных из Web-страницы. Используется, если свойство QueryType установлено в значение xlWebQuery
WebDisable-Date-Recognition	Boolean	Устанавливает/возвращает необходимость разбора данных, выглядящих как дата, при импорте данных из Web-страницы. Используется, если свойство QueryType установлено в значение xlWebQuery
WebDisable-Redirections	Boolean	Устанавливает/возвращает отключение перенаправлений в Web-запросе для объекта QueryTable
Web-Formatting	xlWebFormatting	Устанавливает/возвращает необходимость сохранения форматирования при импорте данных из Web-страницы (например, xlAll, xlNone). Используется, если свойство QueryType установлено в значение xlWebQuery
WebPre-Formatted-TextTo-Columns	Boolean	Устанавливает/возвращает необходимость разбора в столбцы текста из тегов <PRE>. Используется, если свойство QueryType установлено в значение xlWebQuery

Объект Range

Это один из самых разносторонних объектов в Excel. Диапазон может содержать единственную ячейку, столбец, строку, непрерывный блок ячеек или несмежные блоки ячеек. Основным родительским объектом для объекта Range является объект Worksheet. Но большинство объектов из объектной модели Excel обращаются к объекту Range. Свойство Range объекта Worksheet используется для выбора определенного диапазона ячеек с помощью параметров Cell1 и Cell2.

Общие свойства объекта Range

Определения свойств Application, Creator и Parent приводились в начале приложения.

Свойства объекта Range

Имя	*Тип возвращаемого значения*	*Описание*
AddIndent	Variant	Устанавливает возвращает необходимость автоматического выравнивания текста в ячейке, если включено равномерное распределение текста
Address	String	Только чтение. Параметры: RowAbsolute, CoumnAbsolute, ReferenceStyle As XlReferenceStyle, [External], [RelativeTo]. Возвращает адрес текущего диапазона в виде строки на языке макроса. Тип адреса (относительный, абсолютный, в формате A1, в формате R1C1) указывается с помощью параметров
AddressLocal	String	Только чтение. Параметры: RowAbsolute, ColumnAbsolute, ReferenceStyle As XlReferenceStyle, [External], [RelativeTo]. Возвращает адрес текущего диапазона в виде строки на языке пользователя. Тип адреса (относительный, абсолютный, в формате A1, в формате R1C1) указывается с помощью параметров
AllowEdit	Boolean	Только чтение. Возвращает значение True, если диапазон можно редактировать на защищенном листе
Areas	Areas	Только чтение. Возвращает объект, содержащий различные несмежные диапазоны в пределах текущего диапазона
Borders	Borders	Только чтение. Возвращает отдельные границы вокруг диапазона. Каждую границу можно извлечь из коллекции по отдельности
Cells	Range	Только чтение. Возвращает ячейки из текущего диапазона. Свойство Cells возвращает диапазон, совпадающий с текущим
Characters	Characters	Только чтение. Параметры: [Start], [Length]. Возвращает все символы из текущего диапазона
Column	Long	Только чтение. Возвращает номер первого столбца диапазона
Columns	Range	Только чтение. Возвращает диапазон столбцов текущего диапазона

Имя	*Тип возвращаемого значения*	*Описание*
ColumnWidth	Variant	Устанавливает/возвращает ширину всех столбцов диапазона. Возвращает значение Null, если столбцы диапазона имеют разную ширину
Comment	Comment	Только чтение. Возвращает объект, соответствующий комментарию к диапазону
Count	Long	Только чтение. Возвращает количество ячеек в текущем диапазоне
CurrentArray	Range	Только чтение. Возвращает объект Range, который соответствует массиву, связанному с определенным диапазоном ячеек, если ячейка является частью массива
CurrentRegion	Range	Только чтение. Возвращает текущий диапазон, в котором содержится объект Range. Диапазон определяется как область, окруженная пустыми ячейками
Dependents	Range	Только чтение. Возвращает зависящие от ячейки объекты в пределах листа
DirectDependents	Range	Только чтение. Возвращает непосредственно зависящие от ячейки объекты в пределах листа
DirectPrecedents	Range	Только чтение. Возвращает объекты в пределах листа, от которых зависит ячейка
End	Range	Только чтение. Параметры: Direction As XlDirection. Возвращает ячейку в конце диапазона, содержащего объект Range. Направление к концу диапазона определяется параметром Direction
EntireColumn	Range	Только чтение. Возвращает полные столбцы листа, занятые текущим диапазоном
EntireRow	Range	Возвращает полные строки листа, занятые текущим диапазоном
Errors	Errors	Только чтение. Возвращает коллекцию Errors, связанную с объектом Range
Font	Font	Только чтение. Возвращает объект с параметрами шрифта для текста в диапазоне
FormatConditions	FormatConditions	Только чтение. Возвращает объект, в котором содержатся параметры условного форматирования текущего диапазона
Formula	Variant	Устанавливает/возвращает формулу для ячеек диапазона
FormulaArray	Variant	Устанавливает/возвращает формулу массива ячеек диапазона
FormulaHidden	Variant	Устанавливает/возвращает необходимость сокрытия формулы при защите книги/листа
FormulaLabel	XlFormulaLabel	Устанавливает/возвращает тип метки формулы, которая используется в указанном диапазоне
FormulaLocal	Variant	Устанавливает/возвращает формулу диапазона на языке пользователя с применением ссылок в формате A1
FormulaR1C1	Variant	Устанавливает/возвращает формулу диапазона на языке макроса с использованием формата ссылок R1C1
FormulaR1C1Local	Variant	Устанавливает/возвращает формулу диапазона на языке пользователя с применением ссылок в формате R1C1

Имя	*Тип возвращаемого значения*	*Описание*
HasArray	Variant	Только чтение. Возвращает информацию о том, является ли состоящий из одной ячейки диапазон частью формулы массива
HasFormula	Variant	Только чтение. Возвращает наличие формул во всех ячейках диапазона (True). Если формулы присутствуют только в некоторых ячейках, свойство возвращает значение Null
Height	Variant	Только чтение. Возвращает высоту диапазона
Hidden	Variant	Устанавливает/возвращает необходимость сокрытия ячеек в диапазоне. Работает, если диапазон включает в себя полные строки или столбцы
Horizontal-Alignment	Variant	Устанавливает/возвращает параметры горизонтального выравнивания ячеек. Для установки свойства используются константы XLHAlign
Hyperlinks	Hyperlinks	Только чтение. Возвращает коллекцию гиперссылок, находящихся в диапазоне
ID	String	Устанавливает/возвращает значение параметра ID для диапазона, если лист сохраняется в виде Web-страницы
IndentLevel	Variant	Устанавливает/возвращает уровень смещения влево для диапазона
Interior	Interior	Только чтение. Возвращает объект, содержащий параметры форматирования внутренней области диапазона (например, цвет внутренней области)
Left	Variant	Только чтение. Возвращает расстояние от левого края самого левого столбца диапазона до левого края столбца A
ListHeader-Rows	Long	Только чтение. Возвращает количество строк заголовков в диапазоне
ListObjects	Range	Возвращает объект Range
Location-Table	XlLocationIn-Table	Только чтение. Возвращает расположение верхнего левого угла диапазона
Locked	Variant	Устанавливает/возвращает возможность модификации ячеек при защите листа. Возвращает значение Null, если некоторые ячейки диапазона заблокированы
MergeArea	Range	Только чтение. Возвращает диапазон, содержащий объединенный диапазон текущего диапазона ячеек
MergeCells	Variant	Устанавливает/возвращает присутствие в текущем диапазоне объединенных ячеек
Name	Variant	Устанавливает/возвращает объект Name, в котором содержится имя диапазона
Next	Range	Только чтение. Возвращает следующий диапазон на листе
NumberFormat	Variant	Устанавливает/возвращает формат чисел, связанный ячейками диапазона. Если в ячейках используется разный формат чисел, свойство возвращает значение Null
Number-FormatLocal	Variant	Устанавливает/возвращает формат чисел, связанный ячейками диапазона. Формат возвращается на языке пользователя. Если в ячейках используется разный формат чисел, свойство возвращает значение Null

Имя	Тип возвращаемого значения	Описание
Offset	Range	Только чтение. Параметры: [RowOffset], [ColumnOffset]. Возвращает ячейку в виде объекта Range. Расположение возвращаемой ячейки относительно текущей ячейки определяется параметрами. Положительное значение параметра RowOffset соответствует смещению вниз. Отрицательное значение параметра RowOffset соответствует смещению вверх. Положительное значение параметра ColumnOffset соответствует смещению вправо. Отрицательное значение параметра ColumnOffset соответствует смещению влево
Orientation	Variant	Устанавливает/возвращает ориентацию текста в ячейке. Значение может находиться в диапазоне от -90 до 90 градусов. Вместо этого для установки свойства могут использоваться константы XlOrientation
OutlineLevel	Variant	Устанавливает/возвращает уровень сокрытия для диапазона строки или столбца
PageBreak	Long	Устанавливает/возвращает разрыв строки, расположенный в диапазоне. Для установки свойства используются константы XlPageBreak
Phonetic	Phonetic	Только чтение. Возвращает объект Phonetic, связанный с диапазоном ячеек
Phonetics	Phonetics	Только чтение. Возвращает объекты Phonetic из диапазона
PivotCell	PivotCell	Только чтение. Возвращает объект PivotCell, который соответствует ячейке из отчета сводной таблицы
PivotField	PivotField	Только чтение. Возвращает поле сводной таблицы, связанное с верхним левым углом текущего диапазона
PivotItem	PivotItem	Только чтение. Возвращает элемент сводной таблицы, связанный с верхним левым углом текущего диапазона
PivotTable	PivotTable	Только чтение. Возвращает отчет сводной таблицы, связанный с верхним левым углом текущего диапазона
Precedents	Range	Только чтение. Возвращает диапазон расположенных на листе ячеек, от которых зависит текущий диапазон ячеек
Prefix-Character	Variant	Только чтение. Возвращает символ, который использовался для определения типа данных в диапазоне ячеек. Например, для текстовой метки используется символ '''
Previous	Range	Только чтение. Возвращает предыдущий диапазон на листе
QueryTable	QueryTable	Только чтение. Возвращает таблицу запроса, связанную с верхним левым углом текущего диапазона
Range	Range	Только чтение. Параметры: Cell1, [Cell2]. Возвращает объект Range, определяемый параметрами Cell1 и Cell2. В параметрах используются относительные ссылки на ячейки. Например, вызов Range.Range("A1") возвращает первый столбец родительского диапазона, но это не обязательно первый столбец листа
ReadingOrder	Long	Устанавливает/возвращает направление текста: справа налево (xlRTL), слева направо (xlLTR) или в зависимости от контекста (xlContext)

Имя	*Тип возвращаемого значения*	*Описание*
Resize	Range	Только чтение. Параметры: [RowSize], [ColumnSize]. Возвращает диапазон указанного размера. Размер указывается с помощью свойств RowSize и ColumnSize
Row	Long	Только чтение. Возвращает номер строки, в которой расположена первая строка диапазона
RowHeight	Variant	Устанавливает/возвращает высоту строк в диапазоне. Если строки диапазона имеют разную высоту, свойство возвращает значение Null
ShowDetail	Variant	Устанавливает/возвращает необходимость развертывания всех уровней диапазона. Используется только с диапазонами, которые являются итоговыми столбцом или строкой
ShrinkToFit	Variant	Устанавливает/возвращает необходимость автоматического уменьшения размера шрифта в соответствии с шириной столбца. Если ячейки в диапазоне имеют различные значения свойства ShrinkToFit, свойство возвращает значение Null
SmartTags	SmartTags	Только чтение. Возвращает объект SmartTag, соответствующий идентификатору в указанной ячейке
SoundNote	SoundNote	Существует для сохранения обратной совместимости
Style	Variant	Только чтение. Возвращает объект Style, связанный с диапазоном
Summary	Variant	Только чтение. Возвращает информацию о том, является ли диапазон итоговой строкой или итоговым столбцом
Text	Variant	Только чтение. Возвращает текст ячейки диапазона
Top	Variant	Только чтение. Возвращает расстояние от верхнего края верхней строки диапазона до верхнего края строки A
UseStandard-Height	Variant	Устанавливает/возвращает необходимость использования стандартной высоты строк. Если строки в пределах диапазона имеют разную высоту, свойство возвращает значение Null
UseStandard-Width	Variant	Устанавливает/возвращает необходимость использования стандартной ширины столбцов. Если столбцы в пределах диапазона имеют разную ширину, свойство возвращает значение Null
Validation	Validation	Только чтение. Возвращает результат проверки данных для текущего диапазона
Value	Variant	Параметры: [RangeValueDataType]. Устанавливает/возвращает значение ячейки или массива ячеек в зависимости от содержимого объекта Range
Value2	Variant	Устанавливает/возвращает значение ячейки или массива ячеек в зависимости от содержимого объекта Range. Свойство Value2 не возвращает типы данных Date и Currency
Vertical-Alignment	Variant	Устанавливает/возвращает вертикальное выравнивание ячеек в диапазоне. Для установки этого свойства используются константы XLVAlign
Width	Variant	Только чтение. Возвращает ширину диапазона

Имя	*Тип возвращаемого значения*	*Описание*
Worksheet	Worksheet	Только чтение. Возвращает лист, в котором хранится объект Range
WrapText	Variant	Устанавливает/возвращает необходимость переноса текста внутри ячейки. Возвращает значение Null, если ячейки в диапазоне содержат различные свойства переноса текста
XPath	Object	Предоставляет доступ к элементу XPath объекта в текущем диапазоне

Методы объекта Range

Имя	*Тип возвращаемого значения*	*Параметры*	*Описание*
Activate	Variant		Выделяет диапазон ячеек
AddComment	Comment	[Text]	Добавляет текст из параметра в ячейку диапазона. Диапазон должен состоять из одной ячейки
AdvancedFilter	Variant	Action As XlFilterAction, [CriteriaRange], [CopyToRange], [Unique]	Копирует или фильтрует данные о текущем диапазоне. Параметр Action указывает операцию: копирование или фильтрация. Параметр CriteriaRange описывает диапазон с критериями. Параметр CopyToRange описывает диапазон, куда должны быть скопированы отфильтрованные данные (если параметр Action установлен в значение xlFilterCopy)
ApplyNames	Variant	Names, IgnoreRelativeAbsolute, UseRowColumnNames, OmitColumn, OmitRow, Order As XlApplyNamesOrder, [AppendLast]	Применяет определенные имена к формулам в диапазоне. Например, если в ячейке содержится формула =$A$1*100 и ячейке $A$1 было присвоено имя "TopLeft", его можно применить к диапазону, и формула изменится на =TopLeft*100. Обратите внимание, что метода UnApplyNames не существует
ApplyOutlineStyles	Variant		Применяет к диапазону стили структуры
AutoComplete	String	String As String	Возвращает и пытается автоматически завершить слово, указанное в параметре String. Возвращает целое слово, если оно существует. Возвращает пустую строку, если такого слова не существует или существует более одного слова

Имя	*Тип возвращаемого значения*	*Параметры*	*Описание*
AutoFill	Variant	Destination As Range, Type As XlAutoFillType	Использует текущий диапазон в качестве источника данных для автоматического заполнения диапазона, на который указывает параметр Destination. Кроме этого, параметр Type используется для определения типа заполнения (например, xlFillCopy, xlFillDays)
AutoFilter	Variant	Field, Criteria1, Operator As XlAutoFilterOperator, [Criteria2], [VisibleDropDown]	Создает автофильтр для данных в диапазоне. Дополнительная информация о параметрах приводится в разделе, посвященном объекту AutoFilter
AutoFit	Variant		Меняет ширину столбцов в соответствии с объемом данных в диапазоне. Диапазон должен содержать полные строки или столбцы
AutoFormat	Variant	Format As XlRangeAutoFormat, [Number], [Font], [Alignment], [Border], [Pattern], [Width]	Форматирует диапазон в соответствии с выбранным форматом (параметр Format). Остальные параметры являются индикаторами, включающими форматирование чисел (Number), шрифтов (Font), выравнивания (Alignment), границ (Border) и шаблонов (Pattern). Кроме этого, указывается способ изменения ширины столбцов в соответствии с применяемым форматированием
AutoOutline	Variant		Создает структуру для диапазона
BorderAround	Variant	LineStyle, Weight As XlBorderWeight, ColorIndex As XlColorIndex, [Color]	Создает границу вокруг диапазона с соответствующим стилем линии (LineStyle), толщиной (Weight) и цветом (ColorIndex)
Calculate	Variant		Вычисляет значение всех формул в диапазоне
CheckSpelling	Variant	[CustomDictionary], [IgnoreUppercase], [AlwaysSuggest], [SpellLang]	Проверяет орфографию текста в диапазоне. При этом можно указать собственный словарь (CustomDictionary), отключить проверку слов, состоящих из ЗАГЛАВНЫХ букв (IgnoreUppercase), и включить вывод списка предложенных вариантов замены (AlwaysSuggest)
Clear	Variant		Очищает ячейки диапазона от текста

Имя	*Тип возвращаемого значения*	*Параметры*	*Описание*
ClearComments			Очищает ячейки диапазона от комментариев
ClearContents	Variant		Очищает ячейки диапазона от формул и значений
ClearFormats	Variant		Очищает ячейки диапазона от форматирования
ClearNotes	Variant		Очищает ячейки диапазона от заметок
ClearOutline	Variant		Удаляет outline из текущего диапазона
ColumnDifferences	Range	Comparison	Возвращает диапазон ячеек, которые отличаются от диапазона, передаваемого в качестве значения параметра Comparison
Consolidate	Variant	[Sources], [Function], [TopRow], [LeftColumn], [CreateLinks]	Консолидирует исходный массив строк ссылок на диапазоны из параметра Sources и возвращает результат в текущий диапазон. Параметр Function используется для установки функции консолидации. Для установки этого параметра применяются константы XLConsolidationFunction
Copy	Variant	[Destination]	Копирует текущий диапазон в указанный диапазон или в буфер обмена, если диапазон назначения не указан
CopyFromRecordset	Long	Data As Recordset, [MaxRows], [MaxColumns]	Копирует в текущий диапазон записи из набора записей ADO или DAO, указанного в качестве значения параметра Data. Набор записей не может содержать объекты OLE
CopyPicture	Variant	Appearance As XlPictureAppearance, Format As XlCopyPictureFormat	Копирует изображение диапазона в буфер обмена. Параметр Appearance используется для выбора экранного представления или представления для печати. Параметр Format определяет тип изображения, которое копируется в буфер обмена
CreateNames	Variant	[Top], [Left], [Bottom], [Right]	Создает именованный диапазон для элементов из текущего диапазона. Для отображения имен диапазонов в верхней строке параметр Top устанавливается в значение True. Для отображения имен диапазонов в нижней строке параметр Bottom устанавливается в значение True. Установка параметров Left или Right в значение True приведет к отображению имен в левом или правом столбце соответственно

Имя	*Тип возвращаемого значения*	*Параметры*	*Описание*
Create-Publisher	Variant	Edition, Appearance As XlPictureAppearance, [ContainsPICT], [ContainsBIFF], [ContainsRTF], [ContainsVALU]	Создает публикатор на основе диапазона. Метод доступен только на платформе Macintosh в операционной системе System 7 и в более поздних версиях
Cut	Variant	[Destination]	Вырезает текущий диапазон в диапазон, указанный в параметре Destination. Если диапазон назначения не указан, диапазон вырезается в буфер обмена
DataSeries	Variant	Rowcol, Type As XlDataSeriesType, Date As XlDataSeriesDate, [Step], [Stop], [Trend]	Создает последовательность данных на месте текущего диапазона
Delete	Variant	[Shift]	Удаляет ячейки в текущем диапазоне и, при необходимости, смещает ячейки в указанном направлении (направление указывается через параметр Shift). Для установки параметра Shift используются константы XlDeleteShiftDirection
DialogBox	Variant		Выводит диалоговое окно, определенное на листе макросов Excel 4.0
Dirty			Включает пересчет диапазона при следующей операции пересчета
Edition-Options	Variant	Type As XlEditionType, Option As XlEditionOptionsOption, Name, Reference, Appearance As XlPictureAppearance, ChartSize As XlPictureAppearance, [Format]	Используется на платформе Macintosh. Устанавливает реакцию диапазона на использование в качестве источника (публикатора) или цели (подписчика) ссылки. Объекты Editions примерно соответствуют связям DDE на платформе Windows
FillDown	Variant		Копирует содержимое и форматирование из верхней строки в остальные строки диапазона
FillLeft	Variant		Копирует содержимое и форматирование из самого правого столбца в остальные столбцы диапазона
FillUp	Variant		Копирует содержимое и форматирование из самой нижней строки в остальные строки диапазона

Имя	*Тип возвращаемого значения*	*Параметры*	*Описание*
Find	Range	What As Variant, [After], [LookIn], [LookAt], [SearchOrder], [SearchDirection As XlSearchDirection], [MatchCase], [MatchByte], [Search Format]	Просматривает текущий диапазон в поисках текста или типа данных, указанного в качестве значения параметра What. Для выбора начальной позиции параметру After присваивается диапазон, состоящий из одной ячейки. Параметр LookIn определяет область поиска
FindNext	Range	[After]	Находит следующее соответствие критерию поиска, определенное с помощью метода Find
FindPrevious	Range	[After]	Находит предыдущее соответствие критерию поиска, определенное с помощью метода Find
FunctionWizard	Variant		Запускает мастер функций для верхней левой ячейки текущего диапазона
GoalSeek	Boolean	Goal, ChangingCell As Range	Возвращает значение True, если значение параметра Goal возвращается в результате модификации диапазона ячеек ChangingCell
Group	Variant	[Start], [End], [By], [Periods]	Снижает уровень структуры в диапазоне или группирует несмежные диапазоны в текущем объекте Range
Insert	Variant	[Shift], [CopyOrigin]	Вставляет эквивалентные строки или столбцы из диапазона в другой диапазон листа
InsertIndent		InsertAmount As Long	Выравнивает диапазон в соответствии со значением параметра InsertAmount
Justify	Variant		Равномерно распределяет текст в ячейках текущего диапазона
ListNames	Variant		Вставляет все имена в пределах текущего диапазона начиная с верхней левой ячейки диапазона
Merge		[Across]	Объединяет ячейки в диапазоне. Для слияния каждой строки в виде отдельной ячейки параметр Across устанавливается в значение True
NavigateArrow	Variant	[TowardPrecedent], [ArrowNumber], [LinkNumber]	Перемещает стрелки зависимости в книге из текущего диапазона, возвращая диапазон назначения. Стрелки зависимости должны быть включены. Для этого используются методы ShowDependents и ShowPrecendents

Имя	*Тип возвращаемого значения*	*Параметры*	*Описание*
NoteText	String	[Text], [Start], [Length]	Устанавливает/возвращает заметки ячеек из текущего диапазона
Parse	Variant	[ParseLine], [Destination]	Разбирает строку, указанную с помощью параметра ParseLine, и возвращает ее в текущий диапазон, разделенную на столбцы. Параметр Destination определяет диапазон назначения. Строка ParseLine должна иметь формат "[ColumnA] [ColumnB]"
PasteSpecial	Variant	Paste As XlPasteType, Operation As XlPasteSpecialOperation, [SkipBlanks], [Transpose]	Вставляет диапазон из буфера обмена в текущий диапазон. Для выбора вставляемых элементов (например, формулы, значения) используется параметр Paste. Параметр Operation определяет дальнейшие действия над вставляемым диапазоном. Для отказа от копирования пустых ячеек параметр SkipBlanks устанавливается в значение True. Для транспонирования диапазона параметр Transpose также устанавливается в значение True
PrintOut	Variant	[From], [To], [Copies], [Preview], [ActivePrinter], [PrintToFile], [Collate], [PrToFileName]	Печатает диаграммы из коллекции. Параметры определяют принтер, количество копий, порядок сортировки и необходимость предварительного просмотра
PrintPreview	Variant	[EnableChanges]	Включает предварительный просмотр текущего диапазона. Для отключения кнопок Параметры страницы и Поля параметр EnableChanges нужно установить в значение False. В результате пользователь не сможет изменить параметры печати
ReplaceSubtotal	Variant		Удаляет итоговые значения из списка в текущем диапазоне
Replace	Boolean	What As Variant, Replacement As Variant, [LookAt], [SearchOrder], [MatchCase], [MatchByte], [SearchFormat], [ReplaceFormat]	Находит текст, указанный в качестве значения параметра What, и заменяет его на содержимое параметра Replacement. Параметр SearchOrder вместе с константами XLSearchOrder позволяет выбрать поиск в строках и в столбцах

Имя	*Тип возвращаемого значения*	*Параметры*	*Описание*
RowDifferences	Range	Comparison	Возвращает диапазон ячеек, отличающийся от значения параметра Comparison
Run	Variant	[Arg1], [Arg2], ... [Arg30]	Запускает макрос Excel 4.0, расположенный в текущем диапазоне. Потенциальные аргументы макроса передаются через параметры метода
Select	Variant		Выделяет ячейки в диапазоне
SetPhonetic			Создает объект Phonetic для каждой ячейки диапазона
Show	Variant		Прокручивает окно Excel для отображения текущего диапазона. Используется только с диапазонами, которые состоят из одной ячейки
ShowDependents	Variant	[Remove]	Выводит зависимые ячейки текущего диапазона, состоящего из одной ячейки
ShowErrors	Variant		Выводит источник ошибок в текущем диапазоне
ShowPrecedents	Variant	[Remove]	Показывает ячейки, от которых зависит текущий диапазон, состоящий из одной ячейки
Sort	Variant	[Key1], [Order1 As XlSortOrder], [Key2], [Type], [Order2 As XlSortOrder], [Key3], [Order3 As XlSortOrder], [Header As XlYesNoGuess], [OrderCustom], [MatchCase], [Orientation As XlSortOrientation], [SortMethod As XlSortMethod], [DataOption1 As XlSortDataOption], [DataOption2 As XlSortDataOption], [DataOption3 As XlSortDataOption]	Сортирует ячейки в диапазоне. Если диапазон содержит только одну ячейку, выполняется поиск в активном диапазоне. Параметры Key1, Key2 и Key3 определяют столбцы для сортировки. Параметры Order1, Order2 и Order3 устанавливают порядок сортировки. Параметр Header указывает на присутствие заголовков в первой строке. Параметр MatchCase определяет учет регистра при сортировке. Параметр Orientation определяет сортировку строк или столбцов. Наконец, параметр SortMethod определяет метод сортировки для других языков (например, xlStroke или xlPinYin). Метод SortSpecial предназначен для сортировки языков Дальнего Востока

Имя	*Тип возвращаемого значения*	*Параметры*	*Описание*
SortSpecial	Variant	[SortMethod As XlSortMethod], [Key1], [Order1 As XlSortOrder], [Type], [Key2], [Order2 As XlSortOrder], [Key3], [Order3 As XlSortOrder], [Header As XlYesNoGuess], [OrderCustom], [MatchCase], [Orientation As XlSortOrientation], [DataOption1 As XlSortDataOption], [DataOption2 As XlSortDataOption], [DataOption3 As XlSortDataOption]	Сортирует диапазон с использованием методов сортировки для языков Дальнего Востока. Параметры совпадают с параметрами метода Sort
Speak		[SpeakDirection], [SpeakFormulas]	Включает произнесение вслух содержимого диапазона по строкам или столбцам
SpecialCells	Range	Type As XlCellType, [Value]	Возвращает ячейки из текущего диапазона, содержащие специальные атрибуты, которые определены параметром Type. Например, если параметр Type установлен в значение xlCellTypeBlanks, возвращается диапазон, содержащий все пустые ячейки
SubscribeTo	Variant	Edition As String, Format As XlSubscribeToFormat	Метод используется только на платформе Macintosh
Subtotal	Variant	GroupBy As Long, Function As XlConsolidationFunction, TotalList, Replace, PageBreaks, SummaryBelowData As XlSummaryRow	Создает поле итогов для диапазона. Если диапазон состоит из одной ячейки, то ячейка итогов создается для активного диапазона. Параметр GroupBy определяет группируемое поле (для подсчета итогов). Параметр Function описывает группировку полей. Параметр TotalList содержит массив смещений полей, которые описывают суммируемые поля. Для замены существующих итогов параметр Replace устанавливается в значение True. Для добавления разрыва страницы после каждой группы параметр PageBreaks устанавливается в значение True. Параметр SummaryBelowData позволяет указать расположение строки итогов

Имя	*Тип возвращаемого значения*	*Параметры*	*Описание*
Table	Variant	[RowInput], [ColumnInput]	Создает новую таблицу данных в текущем диапазоне
TextTo-Columns	Variant	[Destination], [DataType As XlTextParsingType], [TextQualifier As XlTextQualifier], [ConsecutiveDelimiter], [Tab], [Semicolon], [Comma], [Space], [Other], [OtherChar], [FieldInfo], [DecimalSeparator], [ThousandsSeparator], [TrailingMinus-Numbers]	Разбирает текст в ячейках на несколько столбцов. Параметр Destination определяет точку назначения текста после разбора. Параметр DataType используется для выбора разделителя или фиксированной длины полей. Параметр Text-Qualifier определяет символ определения строковых данных при разборе. Параметр Con-secutiveDelimiter нужно установить в значение True для интерпретации нескольких последовательных разделителей в качестве одного разделителя. Для использования другого разделителя параметр Other устанавливается в значение True и в качестве значения параметра OtherChar передается новый символ разделителя. Параметру FieldInfo присваивается двумерный массив, содержащий дополнительную информацию для разбора. Параметры DecimalSeparator и ThousandsSeparator позволяют определить обработку чисел при разборе
UnGroup	Variant		Повышает уровень структуры в диапазоне или снимает группировку диапазона в отчете сводной таблицы
UnMerge			Разделяет объединенные ячейки

Пример: объект Range

Дополнительная информация и примеры работы с объектом Range приводятся в главе 5.

Объект RecentFile и коллекция RecentFiles

В коллекции RecentFiles хранится список модифицировавшихся в последнее время файлов. Коллекция является эквивалентом списка в меню **Файл** (File) в Excel. Каждый объект RecentFile соответствует одному из модифицированных файлов.

Кроме типичных атрибутов, коллекция RecentFiles предоставляет несколько свойств и метод. Свойство Maximum позволяет установить или получить количество элементов в списке файлов. Значение этого свойства должно находиться в пределах от 0 до 9. Свойство Count возвращает количество объектов RecentFile в коллекции. Метод Add используется для добавления файла в коллекцию (с указанием параметра Name).

Общие свойства объекта RecentFile

Определения свойств Application, Creator и Parent приводились в начале приложения.

Свойства объекта RecentFile

Имя	*Тип возвращаемого значения*	*Описание*
Index	Long	Только чтение. Возвращает положение текущего объекта в коллекции
Name	String	Только чтение. Возвращает имя модифицированного файла
Path	String	Только чтение. Возвращает путь к недавно модифицированному файлу

Методы объекта RecentFile

Имя	*Тип возвращаемого значения*	*Параметры*	*Описание*
Delete			Удаляет объект из коллекции
Open	Workbook		Открывает недавно модифицированный файл и возвращает объект книги

Пример: объект RecentFile и коллекция RecentFiles

```
Sub CheckRecentFiles()
  Dim oRF As RecentFile
  ' Удаляет из списка все файлы, которые ссылаются на
  ' привод гибких дисков
  For Each oRF In Application.RecentFiles
     If Left(oRF.Path, 2) = "A:" Then
        oRF.Delete
     End If
  Next
End Sub
```

Объект RoutingSlip

Объект RoutingSlip предоставляет доступ к свойствам и методам бланка маршрута документа Excel. Родительским для объекта RoutingSlip является объект Workbook. Перед работой с объектом RoutingSlip свойство HasRoutingSlip объекта Workbook должно быть установлено в значение True.

Общие свойства объекта RoutingSlip

Определения свойств Application, Creator и Parent приводились в начале приложения.

Свойства объекта RoutingSlip

Имя	*Тип возвращаемого значения*	*Описание*
Delivery	XlRoutingSlipDelivery	Устанавливает/возвращает способ обработки доставки
Message	Variant	Устанавливает/возвращает текст тела сообщения на бланке маршрута
Recipients	Variant	Параметры: [Index]. Возвращает список имен получателей, которым отправляется родительская книга
ReturnWhenDone	Boolean	Устанавливает/возвращает необходимость отправки уведомления исходному отправителю
Status	XlRoutingSlipStatus	Только чтение. Возвращает текущее состояние бланка маршрута
Subject	Variant	Устанавливает/возвращает текст темы сообщения на бланке маршрута
TrackStatus	Boolean	Устанавливает/возвращает необходимость отправки уведомлений исходному отправителю при каждой пересылке сообщения

Методы объекта RoutingSlip

Имя	*Тип возвращаемого значения*	*Параметры*	*Описание*
Reset	Variant		Сбрасывает бланк маршрута

Объект RTD

Представляет объект данных реального времени, похожий на тот, на который ссылается объект IrtdServer. На момент написания этой книги доступная документация имела минимальный объем.

Свойства объекта RTD

Имя	*Тип возвращаемого значения*	*Описание*
ThrottleInterval	Long	Устанавливает/возвращает интервал времени между обновлениями

Методы объекта RTD

Имя	*Тип возвращаемого значения*	*Параметры*	*Описание*
RefreshData	RTD		Запрашивает у сервера обновление данных реального времени
RestartServers	RTD		Восстанавливает подключение к серверам данных реального времени

Объект Scenario и коллекция Scenarios

В коллекции Scenarios хранится список всех сценариев, связанных с листом. Каждый объект Scenario соответствует одному сценарию, связанному с листом. Сценарий представляет собой список сохраненных значений ячеек, которые могут быть подставлены в ячейки листа. Родительским объектом коллекции Scenarios является объект Worksheet.

Кроме типичных атрибутов, коллекция Scenarios предоставляет несколько свойств и методов. Они перечислены в следующей таблице.

Свойства и методы коллекции Scenarios

Имя	*Тип возвращаемого значения*	*Описание*
Count	Long	Только чтение. Возвращает количество объектов Scenario в коллекции
Add	Scenario	Метод. Параметры: Name As String, ChangingCells, [Values], [Comment], [Locked], [Hidden]. Добавляет сценарий в коллекцию. Параметр Name указывает имя сценария. Описание параметров приводится в разделе, посвященном свойствам объекта Scenario
Create-Summary	Values	Метод. Параметры: ReportType As XlSummaryReportType, [ResultCells]. Создает лист, содержащий итоговую информацию по всем сценариям из родительского листа. Параметр ReportType позволяет определить тип отчета. Параметр ResultCells содержит диапазон ячеек, в которых хранятся формулы для меняющихся ячеек
Merge	Values	Метод. Параметры: Source. Объединяет сценарии из параметра Source на текущем листе

Общие свойства объекта Scenario

Определения свойств Application, Creator и Parent приводились в начале приложения.

Свойства объекта Scenario

Имя	*Тип возвращаемого значения*	*Описание*
Changing-Cells	Range	Только чтение. Возвращает диапазон ячеек на листе, в которые вставляются значения из определенного сценария
Comment	String	Устанавливает/возвращает комментарий сценария
Hidden	Boolean	Устанавливает/возвращает необходимость сокрытия сценария
Locked	Boolean	Устанавливает/возвращает возможность модификации сценария при защите листа
Name	String	Устанавливает/возвращает имя сценария
Values	Variant	Только чтение. Параметры: [Index]. Возвращает массив значений, которые вставляются в ячейки для этого сценария

Методы объекта Scenario

Имя	Тип возвращаемого значения	Параметры	Описание
ChangeScenario	Variant	ChangingCells, [Values]	Меняет набор ячеек на листе, которые меняют значения в соответствии со сценарием. При необходимости можно выбрать новые значения для сценария
Delete	Variant		Удаляет объект Scenario из коллекции
Show	Variant		Отображает результаты сценария, размещая значения сценария на листе

Пример: объект Scenario и коллекция Scenarios

```
Sub GetBestScenario()
   Dim oScen As Scenario
   Dim oBestScen As Scenario
   Dim dBestSoFar As Double
   ' Перебрать сценарии на листе
   For Each oScen In ActiveSheet.Scenarios
      ' Показать сценарий
      oScen.Show

      ' Является ли сценарий лучшим?
      If Range("Result").Value > dBestSoFar Then
         dBestSoFar = Range("Result").Value
            ' Да - сохранить его
         Set oBestScen = oScen
      End If
   Next
   ' Показать лучший сценарий
   oBestScen.Show
   MsgBox "Лучшим сценарием является " & oBestScen.Name
End Sub
```

Объект Series и коллекция SeriesCollection

В коллекции SeriesCollection хранятся ряды из группы диаграммы. Каждый объект Series содержит коллекцию точек для группы диаграммы. Например, простая диаграмма графика содержит ряд (Series) точек, извлеченных из исходных данных. Так как на некоторых диаграммах выполняется построение нескольких рядов точек, эта информация хранится в коллекции SeriesCollection. Родительским объектом для коллекции SeriesCollection является объект ChartGroup.

Кроме типичных атрибутов, коллекция SeriesCollection предоставляет несколько свойств и методов, показанных в следующей таблице.

Свойства и методы коллекции SeriesCollection

Имя	*Тип возвращаемого значения*	*Описание*
Add	Series	Метод. Параметры: Source, Rowcol As XlRowCol, [SeriesLabels], [CategoryLabels], [Replace]. Добавляет в коллекцию объект Series. Параметр Source определяет диапазон или массив с точками данных, описывающими новую последовательность (и все точки в пределах ряда) Параметр Rowcol позволяет выбрать расположение рядов точек в строках или столбцах диапазона из параметра Source. Если первая строка или столбец диапазона из параметра Source содержат метки ряда и категории, параметры SeriesLabels и CategoryLabels устанавливаются в значение True
Count	Long	Только чтение. Возвращает количество объектов Series в коллекции
Extend	Variant	Метод. Параметры: Source, [Rowcol], [CategoryLabels]. Добавляет в коллекцию SeriesCollection точки из диапазона или массива, на которые указывает параметр Source. Дополнительная информация о других параметрах приводится в описании метода Add
Paste	Variant	Метод. Параметры: Rowcol As XlRowCol, [SeriesLabels], [CategoryLabels], [Replace], [NewSeries]. Вставляет данные из буфера обмена в коллекцию SeriesCollection в виде нового объекта Series. Дополнительная информация об остальных параметрах приводится в описании метода Add
NewSeries	Series	Метод. Создает новый ряд и возвращает созданный объект

Общие свойства объекта Series

Определения свойств Application, Creator и Parent приводились в начале приложения.

Свойства объекте Series

Имя	*Тип возвращаемого значения*	*Описание*
ApplyPictToEnd	Boolean	Устанавливает/возвращает необходимость добавления рисунка в конце точек ряда
ApplyPictToFront	Boolean	Устанавливает/возвращает необходимость добавления рисунка в начале точек ряда
ApplyPictToSides	Boolean	Устанавливает/возвращает необходимость добавления рисунка по бокам точек ряда
AxisGroup	XlAxisGroup	Устанавливает/возвращает тип оси, используемой для ряда (основная или дополнительная)
BarShape	XlBarShape	Устанавливает/возвращает тип геометрической фигуры, которая используется в трехмерной диаграмме или гистограмме (например, xlBox)

Имя	*Тип возвращаемого значения*	*Описание*
`Border`	`Border`	Только чтение. Возвращает коллекцию границ (сторон) вокруг ряда. Атрибуты каждой границы могут меняться по отдельности
`BubbleSizes`	`Variant`	Устанавливает/возвращает ссылки на ячейки (в формате A1), содержащие данные о размере пузырьков на пузырьковой диаграмме
`ChartType`	`XlChartType`	Устанавливает/возвращает тип диаграммы, в которой используется ряд
`ErrorBars`	`ErrorBars`	Только чтение. Возвращает полосы ошибки для ряда. Используется вместе со свойством `HasErrorBars`
`Explosion`	`Long`	Устанавливает/возвращает расстояние, на которое доли (точки) круговой или кольцевой диаграммы выдвигаются наружу. Для отказа от выдвижения нужно установить свойство в значение 0
`Fill`	`ChartFillFormat`	Только чтение. Возвращает объект с параметрами форматирования заливки для ряда точек на диаграмме
`Formula`	`String`	Устанавливает/возвращает тип метки формулы, которая используется для ряда
`FormulaLocal`	`String`	Устанавливает/возвращает формулу ряда на языке пользователя с использованием формата ссылок A1
`FormulaR1C1`	`String`	Устанавливает/возвращает формулу ряда на языке макроса с применением формата ссылок R1C1
`FormulaR1C1Local`	`String`	Устанавливает/возвращает формулу ряда на языке пользователя с применением формата ссылок R1C1
`Has3DEffect`	`Boolean`	Устанавливает/возвращает необходимость использования трехмерных эффектов для пузырьковой диаграммы
`HasDataLabels`	`Boolean`	Устанавливает/возвращает необходимость отображения меток данных для точек последовательности
`HasErrorBars`	`Boolean`	Устанавливает/возвращает необходимость отображения полос ошибки для ряда. Используется вместе со свойством `ErrorBars`
`HasLeaderLines`	`Boolean`	Устанавливает/возвращает необходимость отображения линий меток для ряда. Используется вместе со свойством `LeaderLines`
`Interior`	`Interior`	Только чтение. Возвращает объект с параметрами форматирования внутренней области ряда (например, цветом внутренней области)
`InvertIfNegative`	`Boolean`	Устанавливает/возвращает необходимость инвертирования цвета точек ряда для отрицательных значений
`LeaderLines`	`LeaderLines`	Устанавливает/возвращает необходимость отображения линий меток для точек диаграммы
`MarkerBackgroundColor`	`Long`	Устанавливает/возвращает цвет фона маркера точки. Для создания значения цвета используется функция `RGB`

Имя	*Тип возвращаемого значения*	*Описание*
Marker-BackgroundColorIndex	XlColorIndex	Устанавливает/возвращает цвет фона маркера точки. Для установки этого свойства используются константы XlColorIndex или значение индекса цвета в текущей палитре
Marker-ForegroundColor	Long	Устанавливает/возвращает основной цвет маркера точки. Для создания значения цвета используется функция RGB
Marker-ForegroundColorIndex	XlColorIndex	Устанавливает/возвращает основной цвет маркера точки. Для создания значения цвета используются константы XlColorIndex или значение индекса цвета в текущей палитре
MarkerSize	Long	Устанавливает/возвращает размер маркера точки
MarkerStyle	XlMarkerStyle	Устанавливает/возвращает тип маркера, который используется в качестве ключа (например, квадрат, ромб, треугольник, рисунок и т.д.)
Name	String	Устанавливает/возвращает имя ряда
PictureType	XlChart-PictureType	Устанавливает/возвращает способ отображения рисунка возле точки из ряда (например, с растягиванием, в виде мозаики). Для установки этого свойства используются константы XlPictureType
PictureUnit	Long	Устанавливает/возвращает количество единиц, которое представляет рисунок, если свойство PictureType установлено в значение xlScale
PlotOrder	Long	Устанавливает/возвращает порядок построения для этого ряда в коллекции SeriesCollection
Shadow	Boolean	Устанавливает/возвращает необходимость отображения эффекта тени для точек ряда
Smooth	Boolean	Устанавливает/возвращает необходимость сглаживания кривых на диаграммах графика и рассеивания
Type	Long	Устанавливает/возвращает тип ряда
Values	Variant	Устанавливает/возвращает диапазон, в котором хранятся значения ряда или массив фиксированных значений, содержащий значения ряда
XValues	Variant	Устанавливает/возвращает массив значений по оси X, хранящийся в диапазоне или в массиве фиксированных значений

Методы объекта Series

Имя	*Тип возвращаемого значения*	*Параметры*	*Описание*
Apply-CustomType		ChartType As XlChartType	Меняет тип диаграммы в соответствии со значением параметра ChartType

Имя	*Тип возвращаемого значения*	*Параметры*	*Описание*
ApplyDataLabels	Variant	[Type As XlDataLabelsType], [LegendKey], [AutoText], [HasLeaderLines], [ShowSeriesName], [ShowCategoryName], [ShowValue], [Show Percentage], [ShowBubbleSize], [Separator]	Устанавливает свойства меток данных в рядах, указанные с помощью параметров. Параметр Type определяет необходимость сокрытия метки или отображения значения, процента от целого или метки категории. Возле точки может отображаться ключ легенды. Для этого параметр LegendKey необходимо установить в значение True. Для добавления линий меток параметр HasLeaderLines устанавливается в значение True
ClearFormats	Variant		Удаляет форматирование ряда
Copy	Variant		Копирует ряды в буфер обмена
DataLables	Object	[Index]	Возвращает коллекцию меток данных ряда. Параметр Index позволяет получить одну метку данных
Delete	Variant		Удаляет ряд из коллекции
ErrorBar	Variant	[Direction As XlErrorBarDirection], [Include As XlErrorBarInclude], [Type As XlErrorBarType], [Amount], [MinusValues]	Добавляет в ряд полосы ошибки. Параметр Direction позволяет выбрать отображения полосы на оси X или Y. Параметр Include определяет включаемые компоненты ошибки. Параметр Type определяет тип используемых полос ошибки. Параметр Amount определяет сумму ошибки. Параметр MinusValues принимает отрицательную сумму ошибки, если параметр Type установлен в значение xlErrorBarTypeCustom
Paste	Variant		Использует рисунок из буфера обмена в качестве маркера для точек последовательности
Points		[Index]	Возвращает коллекцию точек, связанных с последовательностью, или одну точку, если указано значение параметра Index
Select	Variant		Выделяет точки последовательности на диаграмме
Trendlines		[Index]	Возвращает коллекцию линий тренда, связанных с последовательностью, или одну линию тренда, если указано значения параметра Index

Пример: объект Series и коллекция SeriesCollection

Дополнительная информация и пример использования объекта Series приводится в разделе, посвященном объекту DataLabel.

Объект SeriesLines

Объект SeriesLines позволяет получать доступ к линиям рядов, связывающим значения данных в пределах каждого ряда. Этот объект относится только к группам двумерных диаграмм с областями или гистограмм. Родительским для объекта SeriesLines является объект ChartGroup.

Общие свойства объекта SeriesLines

Определения свойств Application, Creator и Parent приводились в начале приложения.

Свойства объекта SeriesLines

Имя	*Тип возвращаемого значения*	*Описание*
Border	Border	Только чтение. Возвращает свойства границы вокруг линий ряда
Name	String	Только чтение. Возвращает имя объекта SeriesLines

Методы объекта SeriesLines

Имя	*Тип возвращаемого значения*	*Параметры*	*Описание*
Delete	Variant		Удаляет объект SeriesLines
Select	Variant		Выделяет линии ряда на диаграмме

Пример: объект SeriesLines

```
Sub FormatSeriesLines()
    Dim oCG As ChartGroup
    Dim oSL As SeriesLines
    ' Перебрать группы столбцов диаграммы
    For Each oCG In Charts(1).ColumnGroups
       ' Убедиться в существовании линий ряда
       oCG.HasSeriesLines = True
       ' Получить линии последовательности
       Set oSL = oCG.SeriesLines
       ' Отформатировать линии
       With oSL
          .Border.Weight = xlThin
          .Border.ColorIndex = 5
       End With
   Next
End Sub
```

Объект ShadowFormat

Объект ShadowFormat поддерживает управление свойствами форматирования эффекта тени для родительского объекта Shape. Для получения доступа к объекту ShadowFormat нужно воспользоваться свойством Shadow объекта Shape.

Общие свойства ShadowFormat

Определения свойств `Application`, `Creator` и `Parent` приводились в начале приложения.

Свойства объекта ShadowFormat

Имя	*Тип возвращаемого значения*	*Описание*
`ForeColor`	`ColorFormat`	Только чтение. Поддерживает управление основным цветом тени
`Obscured`	`MsoTriState`	Устанавливает/возвращает перекрытие тени геометрической фигурой
`OffsetX`	`Single`	Устанавливает/возвращает горизонтальное смещение тени
`OffsetY`	`Single`	Устанавливает/возвращает вертикальное смещение тени
`Transparency`	`Single`	Устанавливает/возвращает прозрачность тени (от 0 до 1, где 1 соответствует полной прозрачности)
`Type`	`MsoShadowType`	Устанавливает/возвращает тип тени
`Visible`	`MsoTriState`	Устанавливает/возвращает необходимость отображения тени

Методы объекта ShadowFormat

Имя	*Тип возвращаемого значения*	*Параметры*	*Описание*
`IncrementOffsetX`		`Increment As Single`	Меняет горизонтальное смещение тени
`IncrementOffsetY`		`Increment As Single`	Меняет вертикальное смещение тени

Пример: объект ShadowFormat

```
Sub AddShadow()
   Dim oSF As ShadowFormat
   Set oSF = ActiveSheet.Shapes.Range(1).Shadow
   With oSF
       .Type = msoShadow6
       .OffsetX = 5
       .OffsetY = 5
       .ForeColor.SchemeColor = 2
       .Visible = True
   End With
End Sub
```

Объект Shape и коллекция Shapes

В коллекции `Shapes` хранится список геометрических фигур на листе. Объект `Shape` соответствует одной геометрической фигуре, например `AutoShape`, фигуре произвольной формы, объекту OLE (например, изображению), элементу управления ActiveX или рисунку. В качестве родительских объектов коллекции `Shapes` могут выступать объекты `Worksheet` и объект `Chart`.

Кроме типичных атрибутов, коллекция `Shapes` предоставляет несколько методов и свойств, которые рассматриваются в следующей таблице.

Свойства и методы коллекции Shapes

Имя	*Тип возвращаемого значения*	*Описание*
Count	Long	Только чтение. Возвращает количество фигур в коллекции
Range	ShapeRange	Только чтение. Параметры: Index. Возвращает объект ShapeRange, содержащий часть геометрических фигур из коллекции Shapes
AddCallout	Shape	Метод. Параметры: Type As MsoCalloutType, Left As Single, Top As Single, Width As Single, Height As Single. Добавляет в коллекцию выноску
AddConnector	Shape	Метод. Параметры: Type As MsoConnectorType, BeginX As Single, BeginY As Single, EndX As Single, EndY As Single. Добавляет в коллекцию соединитель
AddCurve	Shape	Метод. Параметры: SafeArrayOfPoints. Добавляет в коллекцию кривую Безье
AddDiagram	Shape	Метод. Параметры: Type As MsoDiagramType, Left As Single, Top As Single, Width As Single, Height As Single. Добавляет на лист объект AutoShape
AddFormControl	Shape	Метод. Параметры: Type As XlFormControl, Left As Long, Top As Long, Width As Long, Height As Long. Добавляет в коллекцию элемент управления Excel
AddLabel	Shape	Метод. Параметры: Orientation As MsoTextOrientation, Left As Single, Top As Single, Width As Single, Height As Single. Добавляет в коллекцию метку
AddLine	Shape	Метод. Параметры: BeginX As Single, BeginY As Single, EndX As Single, EndY As Single. Добавляет в коллекцию линию
AddOLEObject	Shape	Метод. Параметры: [ClassType], [Filename], [Link], [DisplayAsIcon], [IconFileName], [IconIndex], [IconLabel], [Left], [Top], [Width], [Height]. Добавляет в коллекцию элемент управления OLE
AddPicture	Shape	Метод. Параметры: Filename As String, LinkToFile As MsoTriState, SaveWithDocument As MsoTriState, Left As Single, Top As Single, Width As Single, Height As Single. Добавляет в коллекцию рисунок
AddPolyline	Shape	Метод. Параметры: SafeArrayOfPoints. Добавляет в коллекцию ломаную линию или замкнутый многоугольник
AddShape	Shape	Метод. Параметры: Type As MsoAutoShapeType, Left As Single, Top As Single, Width As Single, Height As Single. Добавляет в коллекцию фигуру, соответствующую значению параметра Type
AddTextbox	Shape	Метод. Параметры: Orientation As MsoTextOrientation, Left As Single, Top As Single, Width As Single, Height As Single. Добавляет в коллекцию текстовое поле

Имя	Тип возвращаемого значения	Описание
AddText-Effect	Shape	Метод. Параметры: PresetTextEffect As MsoPreset-TextEffect, Text As String, FontName As String, FontSize As Single, FontBold As MsoTriState, FontItalic As MsoTriState, Left As Single, Top As Single. Добавляет в коллекцию объект WordArt
Build-Freeform	Freeform-Builder	Метод. Параметры: EditingType As MsoEditingType, X1 As Single, Y1 As Single. Предоставляет доступ к объекту, поддерживающему создание новой фигуры на основе объекта ShapeNode
SelectAll		Метод. Выделяет все фигуры в коллекции

Общие свойства объекта Shape

Определения свойств Application, Creator и Parent приводились в начале приложения.

Свойства объекта Shape

Имя	Тип возвращаемого значения	Описание
Adjustments	Adjustments	Только чтение. Возвращает объект, обеспечивающий доступ к изменению размеров геометрической фигуры
Alternative-Text	String	Устанавливает/возвращает альтернативный текст, который отображается при невозможности загрузки изображения. Используется вместе с Web-страницами
AutoShape-Type	MsoAutoShape-Type	Устанавливает/возвращает тип объекта AutoShape
BlackWhite-Mode	MsoBlack-WhiteMode	Свойство используется для совместимости с другими пакетами рисования. Полезных функций не имеет
Bottom-RightCell	Range	Только чтение. Возвращает диапазон, состоящий из одной ячейки и соответствующий ячейке под нижним правым углом геометрической фигуры
Callout	CalloutFormat	Только чтение. Возвращает объект, предоставляющий доступ к свойствам выноски
Child	MsoTriState	Только чтение. Возвращает, является ли текущая геометрическая фигура дочерней или имеют ли все фигуры в пределах диапазона одну родительскую фигуру
Connection-SiteCount	Long	Только чтение. Возвращает количество потенциальных точек подключения на геометрической фигуре
Connector	MsoTriState	Только чтение. Возвращает информацию о том, является ли фигура соединителем
Connector-Format	Connector-Format	Только чтение. Возвращает объект, содержащий параметры форматирования соединителя. Используется только вместе с фигурами-соединителями
Control-Format	ControlFormat	Только чтение. Возвращает объект, содержащий параметры форматирования элемента управления Excel
Diagram	Diagram	Только чтение. Возвращает объект Diagram
DiagramNode	DiagramNode	Только чтение. Возвращает узел на диаграмме

Имя	*Тип возвращаемого значения*	*Описание*
`Fill`	`FillFormat`	Только чтение. Возвращает объект, содержащий параметры форматирования заливки для объекта `Shape`
`FormControl-Type`	`XlFormControl`	Только чтение. Возвращает тип элемента управления, которым является текущая фигура (например, `xlCheckBox`). Фигура должна быть элементом управления Excel
`GroupItems`	`GroupShapes`	Только чтение. Возвращает фигуры, из которых состоит текущая фигура
`HasDiagram`	`MsoTriState`	Только чтение. Возвращает наличие диаграммы в фигуре или в диапазоне фигур
`HasDiagram-Node`	`MsoTriState`	Только чтение. Возвращает существование узла диаграммы в фигуре или в диапазоне фигур
`Height`	`Single`	Устанавливает/возвращает высоту геометрической фигуры
`Horizontal-Flip`	`MsoTriState`	Только чтение. Возвращает информацию о том, выполнялся ли горизонтальный поворот фигуры
`Hyperlink`	`Hyperlink`	Только чтение. Возвращает гиперссылку для фигуры
`ID`	`Long`	Только чтение. Возвращает тип указанного объекта
`Left`	`Single`	Устанавливает/возвращает горизонтальное положение геометрической фигуры
`Line`	`LineFormat`	Только чтение. Возвращает объект с параметрами форматирования линии геометрической фигуры
`LinkFormat`	`LinkFormat`	Только чтение. Возвращает объект с параметрами связи OLE
`LockAspect-Ratio`	`MsoTriState`	Устанавливает/возвращает необходимость сохранения соотношения между размерами геометрической формы при изменении размера
`Locked`	`Boolean`	Устанавливает/возвращает возможность модификации фигуры при блокировании листа (значение `True` соответствует запрету на модификацию)
`Name`	`String`	Устанавливает/возвращает имя объекта `Shape`
`Nodes`	`ShapeNodes`	Только чтение. Возвращает объект с узлами фигуры произвольной формы
`OLEFormat`	`OLEFormat`	Только чтение. Возвращает объект, предоставляющий доступ к свойствам объекта OLE
`OnAction`	`String`	Устанавливает/возвращает макрос, который необходимо запустить при щелчке на геометрической фигуре
`ParentGroup`	`Shape`	Только чтение. Возвращает общую родительскую геометрическую фигуру для дочерней геометрической фигуры или диапазона дочерних геометрических фигур
`Picture-Format`	`PictureFormat`	Только чтение. Возвращает объект, предоставляющий доступ к параметрам форматирования рисунка
`Placement`	`XlPlacement`	Устанавливает/возвращает взаимодействие объекта на ячейки вокруг
`Rotation`	`Single`	Устанавливает/возвращает поворот фигуры в градусах
`Script`	`Script`	Только чтение. Возвращает сценарий `VBScript`, связанный с геометрической фигурой

Имя	*Тип возвращаемого значения*	*Описание*
Shadow	ShadowFormat	Только чтение. Возвращает объект, предоставляющий доступ к параметрам форматирования тени
TextEffect	TextEffect-Format	Только чтение. Возвращает объект, предоставляющий доступ к параметрам эффектов текста
TextFrame	TextFrame	Только чтение. Возвращает объект, предоставляющий доступ к параметрам текстового фрейма
ThreeD	ThreeDFormat	Только чтение. Возвращает объект, предоставляющий доступ к параметрам форматирования трехмерных эффектов
Top	Single	Устанавливает/возвращает вертикальное положение фигуры
TopLeftCell	Range	Только чтение. Возвращает диапазон, состоящий из одной ячейки и соответствующий положению верхнего левого угла геометрической фигуры
Type	MsoShapeType	Только чтение. Возвращает тип геометрической фигуры
VerticalFlip	MsoTriState	Только чтение. Возвращает информацию о том, выполнялся ли вертикальный поворот фигуры
Vertices	Variant	Только чтение. Возвращает последовательность пар координат, описывающих вершины фигуры произвольной формы
Width	Single	Только чтение. Возвращает ширину геометрической фигуры
ZOrder-Position	Long	Только чтение. Возвращает положение фигуры относительно других фигур в коллекции (например, на переднем плане, на заднем плане)

Методы объекта Shape

Имя	*Тип возвращаемого значения*	*Параметры*	*Описание*
Apply			Активизирует геометрическую фигуру
Copy			Копирует геометрическую фигуру в буфер обмена
CopyPicture		[Appearance As XLPicture-Appearance], [Format As XlCopyPicture-Format]	Копирует изображение диапазона в буфер обмена. Параметр Appearance используется для выбора экранного представления или представления для печати. Параметр Format определяет тип изображения, копируемого в буфер обмена
Cut			Вырезает геометрическую фигуру и помещает ее в буфер обмена
Duplicate	Shape		Дублирует геометрическую фигуру, возвращая дубликат
Increment-Left		Increment As Single	Перемещает геометрическую фигуру по горизонтали

Имя	*Тип возвращаемого значения*	*Параметры*	*Описание*
Increment-Rotation		Increment As Single	Вращает геометрическую фигуру. Поворот в градусах указывается в качестве значения параметра Increment
IncrementTop		Increment As Single	Перемещает геометрическую фигуру по вертикали
PickUp			Копирует формат текущей геометрической фигуры. После этого формат можно применить к другой геометрической фигуре
Reroute-Connections			Оптимизирует маршрут текущей геометрической фигуры-соединителя между двумя геометрическими фигурами. Кроме этого, метод может использоваться для оптимизации всех соединителей, связанных с текущей геометрической фигурой
ScaleHeight		Factor As Single, Relative-ToOriginalSize As MsoTri-State, [Scale]	Масштабирует высоту геометрической фигуры на значение параметра Factor
ScaleWidth		Factor As Single, Relative-ToOriginalSize As MsoTri-State, [Scale]	Масштабирует ширину геометрической фигуры на значение параметра Factor
Select		[Replace]	Выделяет геометрическую фигуру в документе
Setshapes-Default-Properties			Делает форматирование текущей геометрической фигуры принятым по умолчанию
Ungroup	ShapeRange		Делит объект Shape на составные геометрические фигуры
ZOrder		ZOrderCmd As MsoZOrderCmd	Меняет порядок объектов Shape в коллекции

Пример: объект Shape и коллекция Shapes

Объект Shape является универсальным контейнером для объектов других типов. Примеры использования объекта Shape и коллекции Shapes приводятся в разделах, посвященных соответствующим объектам.

Объект ShapeNode и коллекция ShapeNodes

В коллекции ShapeNodes хранится список узлов и кривых, составляющих геометрическую фигуру произвольной формы. Объект ShapeNode описывает один узел или сегмент кривой. Свойство Nodes объекта Shape используется для получения доступа к коллекции ShapeNodes.

Кроме типичных атрибутов, коллекция ShapeNodes предоставляет несколько методов, описанных в следующей таблице.

Свойства и методы коллекции ShapeNodes

Имя	*Тип возвращаемого значения*	*Описание*
Count	Integer	Только чтение. Возвращает количество объектов ShapeNode в коллекции
Delete		Метод. Параметры: Index As Integer. Удаляет узел, указанный с помощью параметра Index
Insert		Метод. Параметры: Index As Integer, SegmentType As MsoSegmentType, EditingType As MsoEditingType, X1 As Single, Y1 As Single, X2 As Single, Y2 As Single, X3 As Single, Y3 As Single. Вставляет узел или сегмент кривой в коллекцию Nodes
SetEditingType		Метод. Параметры: Index As Integer, EditingType As MsoEditingType. Устанавливает тип редактирования узла
SetPosition		Метод. Параметры: Index As Integer, X1 As Single, Y1 As Single. Перемещает указанный узел
SetSegmentType		Метод. Параметры: Index As Integer, SegmentType As MsoSegmentType. Меняет тип сегмента после узла

Общие свойства объекта ShapeNode

Определения свойств Application, Creator и Parent приводились в начале приложения.

Свойства объекта ShapeNode

Имя	*Тип возвращаемого значения*	*Описание*
EditingType	MsoEditingType	Только чтение. Возвращает тип редактирования узла
Points	Variant	Только чтение. Возвращает пару позиционных координат
SegmentType	MsoSegmentType	Только чтение. Возвращает тип сегмента, следующего за углом

Пример: объект ShapeNode и коллекция ShapeNodes

```
Sub ToggleArch()
   Dim oShp As Shape
   Dim oSN As ShapeNodes
   Set oShp = ActiveSheet.Shapes(1)
   ' Это геометрическая фигура произвольной формы?
   If oShp.Type = msoFreeform Then

      ' Если да, то получить список узлов
      Set oSN = oShp.Nodes
      ' Превратить сегмент 3 из прямой в кривую
      If oSN.Item(3).SegmentType = msoSegmentCurve Then
         oSN.SetSegmentType 3, msoSegmentLine
      Else
         oSN.SetSegmentType 3, msoSegmentCurve
      End If
   End If
End Sub
```

Коллекция ShapeRange

В коллекции ShapeRange хранятся объекты Shape для определенного диапазона или выделения в документе. В качестве возможных родительских объектов могут выступать Range и Selection. Кроме типичных атрибутов, коллекция ShapeRange предоставляет множество свойств и методов, перечисленных в следующей таблице.

Но некоторые операции при использовании вместе с коллекцией ShapeRange, содержащей несколько геометрических фигур, могут приводить к появлению ошибки.

Свойства коллекции ShapeRange

Имя	*Тип возвращаемого значения*	*Описание*
Adjustments	Adjustments	Только чтение. Возвращает объект, поддерживающий изменение формы геометрической фигуры
AlternativeText	String	Устанавливает/возвращает альтернативный текст, который отображается вместо изображения. Используется вместе с Web-страницами
AutoShapeType	MsoAutoShapeType	Устанавливает/возвращает тип используемого объекта AutoShape
BlackWhiteMode	MsoBlackWhiteMode	Применяется для совместимости с другими пакетами рисования. Полезных функций не имеет
Callout	CalloutFormat	Только чтение. Возвращает объект, предоставляющий доступ к свойствам выноски
Child	MsoTriState	Только чтение. Возвращает информацию о том, является ли текущая геометрическая фигура дочерней, или имеют ли все фигуры в пределах диапазона одну родительскую фигуру
ConnectionSiteCount	Long	Только чтение. Возвращает количество потенциальных точек подключения на геометрической фигуре
Connector	MsoTriState	Только чтение. Возвращает информацию о том, является ли фигура соединителем
ConnectorFormat	ConnectorFormat	Только чтение. Возвращает объект, содержащий параметры форматирования соединителя. Используется только вместе с фигурами-соединителями
Count	Long	Только чтение. Возвращает количество геометрических фигур в коллекции
Diagram	Diagram	Только чтение. Возвращает объект Diagram
DiagramNode	DiagramNode	Только чтение. Возвращает узел диаграммы
Fill	FillFormat	Только чтение. Возвращает объект, содержащий свойства заливки для геометрической фигуры
GroupItems	GroupShapes	Только чтение. Возвращает геометрические фигуры, из которых состоит текущая геометрическая фигура
HasDiagram	MsoTriState	Только чтение. Возвращает присутствие диаграммы в пределах объекта Shape или ShapeRange
HasDiagramNode	MsoTriState	Только чтение. Возвращает существование узла диаграммы в объекте Shape или ShapeRange
Height	Single	Устанавливает/возвращает высоту геометрической фигуры

Имя	*Тип возвращаемого значения*	*Описание*
HorizontalFlip	MsoTriState	Только чтение. Возвращает информацию о том, выполнялся ли горизонтальный поворот геометрической фигуры
ID	Long	Только чтение. Возвращает тип указанного объекта
Left	Single	Устанавливает/возвращает горизонтальное положение геометрической фигуры
Line	LineFormat	Только чтение. Возвращает объект с параметрами форматирования линии геометрической фигуры
LockAspectRatio	MsoTriState	Устанавливает/возвращает сохранение соотношения между размерами геометрической фигуры при их изменении
Name	String	Устанавливает/возвращает имя геометрической фигуры
Nodes	ShapeNodes	Только чтение. Возвращает узлы геометрической фигуры
ParentGroup	Shape	Только чтение. Возвращает общую родительскую фигуру дочерней фигуры или диапазона дочерних фигур
PictureFormat	PictureFormat	Только чтение. Возвращает объект с параметрами форматирования рисунка
Rotation	Single	Устанавливает/возвращает угол поворота геометрической фигуры (в градусах)
Shadow	ShadowFormat	Только чтение. Возвращает объект с параметрами эффекта тени
TextEffect	TextEffectFormat	Только чтение. Возвращает объект с параметрами эффектов текста
TextFrame	TextFrame	Только чтение. Возвращает объект с параметрами текстового фрейма
ThreeD	ThreeDFormat	Только чтение. Возвращает объект с параметрами форматирования трехмерных эффектов
Top	Single	Устанавливает/возвращает вертикальное положение геометрической фигуры
Type	MsoShapeType	Только чтение. Возвращает тип геометрической фигуры.
VerticalFlip	MsoTriState	Только чтение. Указывает на выполнение вертикального поворота фигуры
Vertices	Variant	Только чтение. Возвращает последовательность пар координат, описывающих вершины фигуры произвольной формы
Visible	MsoTriState	Устанавливает/возвращает видимость геометрической фигуры
Width	Single	Устанавливает/возвращает ширину геометрической фигуры
ZOrderPosition	Long	Только чтение. Меняет порядок объектов в коллекции

Методы объекта ShapeRange

Имя	*Тип возвращаемого значения*	*Параметры*	*Описание*
Align		AlignCmd As MsoAlignCmd, RelativeTo As MsoTriState	Выравнивает фигуры в коллекции. Критерии выравнивания передаются в качестве значений параметров
Apply			Применяет форматирование, установленное с помощью метода PickUp
Delete			Удаляет геометрическую фигуру
Distribute		DistributeCmd As MsoDistributeCmd, RelativeTo As MsoTriState	Равномерно распределяет фигуры из коллекции по горизонтали или вертикали
Duplicate	ShapeRange		Дублирует фигуру и возвращает новый объект ShapeRange
Flip		FlipCmd As MsoFlipCmd	Выполняет поворот фигуры в соответствии со значением параметра FlipCmd
Group	Shape		Группирует фигуры в коллекции
IncrementLeft		Increment As Single	Перемещает фигуру по горизонтали
IncrementRotation		Increment As Single	Поворачивает фигуру. Угол поворота (в градусах) передается в качестве значения параметра Increment
IncrementTop		Increment As Single	Перемещает фигуру по вертикали
PickUp			Копирует формат текущей фигуры, который можно применять к другим фигурам
Regroup	Shape		Повторно группирует сгруппированные ранее фигуры
RerouteConnections			Оптимизирует маршрут текущей фигуры-соединителя между двумя геометрическими фигурами. Может использоваться для оптимизации всех маршрутов соединителей, связанных с текущей фигурой
ScaleHeight		Factor As Single, RelativeToOriginalSize As MsoTriState, [Scale]	Масштабирует высоту фигуры в соответствии со значением параметра Factor
ScaleWidth		Factor As Single, RelativeToOriginalSize As MsoTriState, [Scale]	Масштабирует ширину фигуры в соответствии со значением параметра Factor
Select		[Replace]	Выделяет фигуру в документе

Имя	Тип возвращаемого значения	Параметры	Описание
SetShapes-Defautl-Properties			Устанавливает форматирование текущей фигуры в качестве принятого по умолчанию
Upgroup	ShapeRange		Разбивает фигуру, которая состоит из объектов Shape
ZOrder		ZOrderCmd As MsoZOrderCmd	Меняет положение объекта Shape в коллекции

Пример: коллекция ShapeRange

```
Sub AlignShapeRanges()
   Dim oSR As ShapeRange
   ' Получить первые две фигуры на листе
   Set oSR = ActiveSheet.Shapes.Range(Array(1, 2))
   ' Выровнять левые края фигур
   oSR.Align msoAlignLefts, msoFalse
End Sub
```

Коллекция Sheets

В коллекции Sheets хранятся все листы родительской книги. В книге могут присутствовать листы диаграмм и листы электронных таблиц, поэтому в коллекции Sheets хранятся объекты Chart и объекты Worksheet из родительской книги. Родительским для коллекции Sheets является объект Workbook.

Общие свойства коллекции Sheets

Определения свойств Application, Creator и Parent приводились в начале приложения

Свойства коллекции Sheets

Имя	Тип возвращаемого значения	Описание
Count	Long	Только чтение. Возвращает количество листов в коллекции (а значит и в книге)
HPageBreaks	HPageBreaks	Только чтение. Возвращает коллекцию горизонтальных разрывов страницы, связанных с коллекцией Sheets
VPageBreaks	VPageBreaks	Только чтение. Возвращает коллекцию вертикальных разрывов страницы, связанных с коллекцией Sheets
Visible	Variant	Устанавливает/возвращает необходимость отображения листов из коллекции. Свойство можно установить в значение xlVeryHidden. В результате пользователь не сможет сделать листы видимыми

Методы коллекции Sheets

Имя	*Тип возвращаемого значения*	*Параметры*	*Описание*
Add		[Before], [After], [Count], [Type]	Добавляет лист в коллекцию. Можно выбрать точку назначения листа, указав объект перед новым объектом листа (параметр Before) или после нового объекта листа (параметр After). Параметр Count определяет количество создаваемых листов. Параметр Type определяет тип листа с помощью констант XLSheetType
Copy		[Before], [After]	Добавляет новую копию активного листа. Положение нового листа определяется параметрами Before и After
Delete			Удаляет все листы в коллекции. Помните, что книга должна иметь как минимум один лист
FillAcross-Sheets		Range As Range, Type As XlFill-With	Копирует значения параметра Range во все остальные листы в тот же диапазон. Параметр Type позволяет выбрать копирование содержимого ячеек, формул или всех компонентов ячеек
Move		[Before], [After]	Перемещает текущий лист на новое место. Новое положение определяется значениями параметров. Дополнительная информация приводилась в описании метода Add
PrintOut		[From], [To], [Copies], [Preview], [ActivePrinter], [PrintToFile], [Collate], [PrToFileName].	Печатает листы в коллекции. Принтер, количество копий, сортировка и необходимость предварительного просмотра определяются значениями параметров. Кроме того, печать листов может быть перенаправлена в файл с помощью параметров PrintToFile и PrToFileName. Параметры From и To определяют диапазон печатаемых страниц
PrintPreview		[EnableChanges]	Выводит текущий лист из коллекции в режиме предварительного просмотра. Установка параметра EnableChanges в значение False отключает кнопки Поля и Параметры страницы. В результате пользователь не сможет модифицировать параметры страницы
Select		[Replace]	Выделяет текущий лист в коллекции

Объект SmartTag и коллекция SmartTags

Объект SmartTag соответствует присвоенному ячейке идентификатору. В составе Excel предоставляется множество встроенных смарт-тегов, например Stock Ticker или распознаватель дат. Кроме того, можно написать собственные смарт-теги на языке Visual Basic. Более подробно смарт-теги рассматриваются в главе 18, но для работы с ними необходимо обладать определенными знаниями по XML.

Коллекция SmartTags описывает все смарт-теги, назначенные ячейкам в пределах приложения.

Общие свойства объекта SmartTag

Определения свойств Application, Creator и Parent приводились в начале приложения.

Свойства объекта SmartTag

Имя	*Тип возвращаемого значения*	*Описание*
DownloadURL	String	Только чтение. Возвращает адрес URL, который сохраняется вместе с соответствующим смарт-тегом
Name	String	Только чтение. Возвращает имя смарт-тега
Properties	CustomProperties	Только чтение. Возвращает свойства смарт-тега
Range	Range	Только чтение. Возвращает диапазон, к которому относится указанный смарт-тег
SmartTagActions	SmartTagActions	Только чтение. Возвращает тип операции для выделенного смарт-тега
XML	String	Только чтение. Возвращает пример кода XML, который передается процедуре с реализацией операции

Методы объекта SmartTag

Имя	*Тип возвращаемого значения*	*Параметры*	*Описание*
Delete			Удаляет объект

Пример: объект SmartTag

Обратите внимание, что этот пример уже рассматривался в главе 18, в разделе “Удаление тега из диапазона”.

Одной из негативных особенностей смарт-тегов являются ложные срабатывания, когда тег помещается в ячейку в результате ошибки. В качестве примера можно привести смарт-тег Stock Ticker, который распознает последовательность “TRUE” в качестве действительного биржевого символа, даже если эта последовательность соответствует значению True типа Boolean. Следующий код обнаруживает все ложные срабатывания и удаляет теги из ячеек:

```
Sub RemoveBooleanTrue()
   Dim oSht As Worksheet
   Dim oTag As SmartTag
   ' Это указатель на смарт-тег StockTicker
   Const sTicker As String = {_}
       "urn:schemas{-}microsoft{-}com:office:smarttags#stockticker"
```

```
    ' Перебрать все листы в активной книге
    For Each oSht In ActiveWorkbook.Worksheets
        ' Перебрать все теги на листе
        For Each oTag In oSht.SmartTags
            ' Это смарт-тег StockTicker, имеющий значение типа Boolean?
            If oTag.Name = sTicker And {_}
                    TypeName(oTag.Range.Value) = "Boolean" Then
                ' Если да, удалить смарт-тег из ячейки
                oTag.Delete
            End If
        Next
    Next
End Sub
```

Объект SmartTagAction и коллекция SmartTagActions

Объект SmartTagAction соответствует операции, поддерживаемой смарт-тегом. Операцией может быть отображение последней цены биржевого символа или назначение встречи на определенную дату.

Коллекция SmartTagActions соответствует всем объектам SmartTagAction в пределах приложения.

Общие свойства объекта SmartTagAction

Определения свойств Application, Creator и Parent приводились в начале приложения.

Свойства объекта SmartTagAction

Имя	*Тип возвращаемого значения*	*Описание*
ActiveXControl	Object	Возвращает ссылку на элемент управления ActiveX, который в данный момент находится в панели задач Document Actions
CheckboxState	Boolean	Возвращает значение True, если флажок установлен. В противном случае свойство возвращает значение False
ExpandHelp	Boolean	Возвращает значение True, если в данный момент развернут элемент управления Smart Document Help. В противном случае возвращается значение False
ListSelection	Long	Возвращает номер пункта в элементе управления List .
Name	String	Только чтение. Возвращает имя смарт-тега
PresentInPane	Boolean	Возвращает значение типа Boolean, которое указывает, отображается ли элемент управления Smart Document в панели задач Document Actions
RadioGroupSelection	Long	Возвращает номер выделенного переключателя в пределах элемента управления RadioGroup
TextboxText	String	Возвращает текст из элемента управления TextBox

Методы объекта SmartTagAction

Имя	*Тип возвращаемого значения*	*Параметры*	*Описание*
Execute			Активизирует операцию смарт-тега

Коллекция SmartTagOptions

Коллекция SmartTagOptions соответствует всем параметрам смарт-тега. Например, в коллекции хранится информация о необходимости внедрения смарт-тега в лист или о необходимости отображения смарт-тегов.

Свойства коллекции SmartTagOptions

Имя	*Тип возвращаемого значения*	*Параметры*	*Описание*
DisplaySmartTags	XlSmartTagDisplayMode		Устанавливает/возвращает параметры отображения смарт-тегов
EmbedSmartTags	Boolean		Устанавливает/возвращает необходимость внедрения смарт-тегов в указанную книгу

Объект SmartTagRecongnizer и коллекция SmartTagRecongnizers

Объект SmartTagRecongnizer предоставляет доступ к механизмам распознавания, которые обрабатывают данные на листе. Эти механизмы могут определяться пользователем, поэтому смарт-теги распознают информацию любого рода. Дополнительная информация приводилась в главе 18.

В коллекции SmartTagRecongnizers хранятся все объекты SmartTagRecongnizer в пределах приложения.

Свойства коллекции SmartTagRecongnizers

Имя	*Тип возвращаемого значения*	*Описание*
Recognize	Boolean	Устанавливает/возвращает возможность распознавания данных с помощью смарт-тегов

Общие свойства объекта SmartTagRecongnizer

Определения свойств Application, Creator и Parent приводились в начале приложения.

Свойства объекта SmartTagRecongnizer

Имя	*Тип возвращаемого значения*	*Описание*
Enabled	Boolean	Устанавливает/возвращает результат распознавания объекта
FullName	String	Только чтение. Возвращает имя объекта, включая путь к файлу
ProgId	String	Только чтение. Возвращает программные идентификаторы объекта

Объект SoundNote

В современных версиях Excel объект SoundNote не используется. Он сохранился в целях обратной совместимости. Список методов этого объекта приводится ниже.

Методы объекта SoundNote

Имя	Тип возвращаемого значения	Параметры
Delete	Variant	
Import	Variant	Filename As String
Play	Variant	
Record	Variant	

Объект Speech

Предоставляет доступ к приложению распознавания речи, которое предоставляется вместе с пакетом Office XP. Новая версия механизма распознавания речи дает возможность чтения вслух текста по требованию или при вводе данных в документ. Для Excel предоставлена возможность чтения содержимого всех ячеек в процессе ввода. Для включения этой возможности воспользуйтесь свойством SpeakCellOnEnter.

Объект Speech доступен из объекта Application.

Свойства объекта Speech

Имя	Тип возвращаемого значения	Описание
Direction	XlSpeakDirection	Устанавливает/возвращает порядок чтения ячеек
SpeakCellOnEnter	Boolean	Устанавливает/возвращает необходимость чтения содержимого активной ячейки при нажатии клавиши <Enter> или при завершении редактирования активной ячейки

Методы объекта Speech

Имя	Тип возвращаемого значения	Параметры	Описание
Speak		Text As String, [SpeakAsync], [SpeakXML], [Purge]	Значение параметра Text читается вслух приложением. Если параметр Purge установлен в значение True, текущая операция чтения будет прервана с удалением из буферов всего текста

Пример: объект Speech

Следующая подпрограмма читает вслух итоговые значения, превышающие установленный минимум (минимум устанавливается в другой ячейке листа).

```
Sub ReadHighExpenses()

    Dim lTotal As Long
    Dim lLimit As Long
    Dim rng As Range

    ' Получить граничное значение
```

```
    lLimit = wksAllowEditRange.Range("Limit")

    ' Перебрать итоговые значения
    For Each rng In wksAllowEditRange.Range("Expenses")
        ' Сохранить текущее итогове значение
        lTotal = rng.Offset(0, 5).Value

        ' Если текущее значение больше установленного предела,
        ' прочитать его вслух
        If lTotal > lLimit Then
            Application.Speech.Speak rng.Text
            Application.Speech.Speak lTotal
        End If
    Next rng

End Sub
```

Коллекция SpellingOptions

Данная коллекция предоставляет доступ к параметрам проверки орфографии в Excel. Эти параметры также доступны на вкладке Орфография (Spelling) в диалоговом окне Сервис⇨Параметры (Tools⇨Options). Коллекция SpellingOptions доступна через объект Application.

Свойства коллекции SpellingOptions

Имя	*Тип возвращаемого значения*	*Описание*
ArabicModes	XlArabicModes	Устанавливает/возвращает режим проверки орфографии арабских языков
DictLang	Long	Устанавливает/возвращает язык словаря, который используется в Excel при проверке орфографии
GermanPostPerform	Boolean	Устанавливает/возвращает проверку орфографии слов с использованием постреформенных правил немецкого языка
HebrewModes	XlHebrewModes	Устанавливает/возвращает режим проверки орфографии еврейских языков
IgnoreCaps	Boolean	Устанавливает/возвращает необходимость проверки слов в верхнем или нижнем регистре
IgnoreFileNames	Boolean	Устанавливает/возвращает необходимость проверки орфографии сетевых адресов Internet или имен файлов
IgnoreMixedDigits	Boolean	Устанавливает/возвращает необходимость проверки орфографии для слов, содержащих цифры
KoreanCombineAux	Boolean	Устанавливает/возвращает необходимость комбинирования вспомогательных глаголов и прилагательных при проверке орфографии для корейского языка
KoreanProcessCompound	Boolean	Устанавливает/возвращает необходимость обработки составных существительных при проверке орфографии для корейского языка
KoreanUseAutoChangeList	Boolean	Устанавливает/возвращает необходимость использования автоматически меняющегося списка слов при проверке орфографии для корейского языка
SuggestMainOnly	Boolean	Устанавливает/возвращает необходимость извлечения предложенных слов только из основного словаря
UserDict	String	Устанавливает/возвращает необходимость создания собственного словаря, в который добавляются новые слова

Пример: коллекция SpellingOptions

Следующая подпрограмма устанавливает параметры проверки орфографии и создает собственный словарь, слова в который добавляются в процессе проверки орфографии.

```
Sub SetSpellingOptions()

    ' Это максимально простой пример
    With Application.SpellingOptions
        .IgnoreCaps = True
        .IgnoreFileNames = True
        .IgnoreMixedDigits = True
        .SuggestMainOnly = False

        ' Это свойство позволяет создать собственный словарь,
        ' который называется Wrox.dic. Словарь можно найти и
        ' отредактировать в каталоге C:\WINDOWS\Application
        ' Data\Microsoft\Proof.
        ' Добавленные во время проверки орфорграфии слова отображжаются в
        ' собственном словаре
        .UserDict = "Wrox.dic"
    End With
```

Объект Style и коллекция Styles

В коллекции `Styles` хранятся определенные пользователем и встроенные стили форматирования, например `Currency` и `Normal`, для книги или диапазона. Каждый объект `Style` предоставляет доступ к атрибутам форматирования, связанным с родительским объектом. Существует несколько встроенных в Excel объектов `Style`, например `Currency`. Кроме этого, можно создавать собственные стили. Возможными родительскими объектами коллекции `Styles` являются объекты `Range` и `Workbook`.

Для доступа к стилям через пользовательский интерфейс можно воспользоваться диалоговым окном Стиль (Style), выбрав команду Формат⇨Стиль (Format⇨Style).

Кроме типичных атрибутов, коллекция `Styles` предоставляет три дополнительных атрибута. Свойство `Count` возвращает количество объектов `Style` в коллекции. Метод `Add` добавляет новый стиль в коллекцию. Параметр `Name` определяет имя нового стиля. Параметр `BasedOn` позволяет выбрать диапазон, на котором будет основан новый стиль. Метод `Merge` объединяет стили текущей книги и книги, на которую указывает параметр `Workbook`.

Общие свойства объекта Style

Определения свойств `Application`, `Creator` и `Parent` приводились в начале приложения.

Свойства объекта Style

Имя	*Тип возвращаемого значения*	*Описание*
`AddIndex`	`Boolean`	Устанавливает/возвращает необходимость автоматического смещения связанного со стилем текста, если для ячейки включено равномерное распределение текста
`Borders`	`Borders`	Только чтение. Возвращает связанную со стилем коллекцию границ. Доступ предоставляется к каждой границе отдельно
`BuiltIn`	`Boolean`	Только чтение. Возвращает информацию о том, является ли стиль встроенным

Имя	*Тип возвращаемого значения*	*Описание*
Font	Font	Только чтение. Возвращает объект, содержащий параметры шрифта для соответствующего стиля
Formula-Hidden	Boolean	Устанавливает/возвращает возможность сокрытия связанных со стилем формул при установке защиты для книги или листа
Horizontal-Alignment	XlHAlign	Устанавливает/возвращает горизонтальное выравнивание для связанных со стилем ячеек. Для установки значения свойства используются константы XlHAlign
Include-Alignment	Boolean	Устанавливает/возвращает включение в стиль свойств, связанных с выравниванием (AddIndent, HorizontalAlignment, VerticalAlignment, WrapText и Orientation)
Include-Border	Boolean	Устанавливает/возвращает включение в стиль атрибутов границ (Color, ColorIndex, LineStyle и Weight)
IncludeFont	Boolean	Устанавливает/возвращает включение в стиль атрибутов шрифта (Background, Bold, Color, ColorIndex, FontStyle, Italic, Name, OutlineFont, Shadow, Size, Strikethrough, Subscript, Superscript и Underline)
Include-Number	Boolean	Устанавливает/возвращает включение в стиль свойства NumberFormat
Include-Patterns	Boolean	Устанавливает/возвращает включение в стиль свойств шаблонов (Color, ColorIndex, InvertIfNegative, Pattern, PatternColor и PatternColorIndex)
Include-Protection	Boolean	Устанавливает/возвращает включение в стиль свойств защиты (FormulaHidden и Locked)
IndentLevel	Long	Устанавливает/возвращает уровень смещения для стиля
Interior	Interior	Только чтение. Возвращает объект, содержащий параметры форматирования внутренней области стиля (например, цвета внутренней области)
Locked	Boolean	Устанавливает/возвращает возможность изменения свойств стиля при включении защиты книги
MergeCells	Variant	Устанавливает/возвращает присутствие в текущем стиле объединенных ячеек
Name	String	Только чтение. Возвращает имя стиля
NameLocal	String	Только чтение. Возвращает имя стиля на языке пользовательского интерфейса
NumberFormat	String	Устанавливает/возвращает связанный со стилем формат чисел
Number-FormatLocal	String	Устанавливает/возвращает связанный со стилем формат чисел на языке пользовательского интерфейса
Orientation	XlOrientation	Устанавливает/возвращает ориентацию текста в связанной со стилем ячейке. Свойство может устанавливаться в значение от -90 до 90 (в градусах). Кроме этого, для установки значения свойства можно использовать константы XlOrientation
ReadingOrder	Long	Устанавливает/возвращает направление чтения текста: справа налево (xlRTL), слева направо (xlLTR) или в зависимости от контекста (xlContext)

Имя	*Тип возвращаемого значения*	*Описание*
ShrinkToFit	Boolean	Устанавливает/возвращает необходимость автоматического изменения ширины связанного со стилем текста в соответствии с шириной столбца
Value	String	Только чтение. Возвращает имя стиля
Vertical-Alignment	XlVAlign	Устанавливает/возвращает вертикальное выравнивание для связанных со стилем ячеек. Для установки свойства используются константы XlVAlign.
WrapText	Boolean	Устанавливает/возвращает перенос текста по словам для связанных со стилем ячеек

Методы объекта Style

Имя	*Тип возвращаемого значения*	*Параметры*	*Описание*
Delete	Variant		Удаляет стиль из коллекции

Пример: объект Style и коллекция Styles

```
Sub UpdateStyles()
   Dim oStyle As Style
   Set oStyle = ActiveWorkbook.Styles("Editing")
   ' Обновить и разблокировать стиль Editing с принятым по умолчанию
   ' цветом фона
   With oStyle
       .IncludePatterns = True
       .IncludeProtection = True
       .Locked = False
       .Interior.Pattern = xlNone
   End With
End Sub
```

Объект Tab

Этот объект соответствует вкладке в нижней части листа диаграммы или листа электронной таблицы. Excel 2003 предоставляет возможность модификации цвета вкладки листа. Для этого используются свойства Color и ColorIndex этого объекта.

Обратите внимание, что установка свойства ColorIndex в значение xlColorIndexAutomatic (это значение отображается в списке автоматического дополнения ввода для свойства) приводит к ошибке.

Общие свойства объекта Tab

Имя	*Тип возвращаемого значения*	*Описание*
Color	Variant	Устанавливает/возвращает основной цвет объекта Tab. Для создания значения цвета используется функция RGB
ColorIndex	XlColorIndex	Устанавливает/возвращает цвет внутренней области

Пример: объект Tab

Следующая подпрограмма меняет цвет вкладки для всех бюджетных листов в соответствии со значением дополнительного свойства каждого листа.

```
Sub ColorBudgetTabs()      Dim bBudget As Boolean
    Dim oCustomProp As CustomProperty
    Dim oCustomProps As CustomProperties
    Dim wks As Worksheet

    ' Перебрать все листы в пределах книги
    For Each wks In ThisWorkbook.Worksheets
        ' Перебрать все дополнительные свойства текущего листа
        ' пока не будет найдено свойство "IsBudget"
        For Each oCustomProp In wks.CustomProperties
            If oCustomProp.Name = "IsBudget" Then
                ' Получить знчение свойства и завершить цикл
                bBudget = CBool(oCustomProp.Value)
                Exit For
            End If
        Next oCustomProp

        ' Воспользоваться значением свойства для
        ' установки цвета вкладки.
        If bBudget Then wks.Tab.ColorIndex = 20 'Light blue

    Next wks

End Sub
```

Объект TextEffectFormat

Объект содержит все свойства и методы, связанные с объектом WordArt. Родительским для объекта TextEffectFormat всегда является объект Shape.

Общие свойства объекта TextEffectFormat

Определения свойств Application, Creator и Parent приводились в начале приложения.

Свойства объекта TextEffectFormat

Имя	*Тип возвращаемого значения*	*Описание*
Alignment	MsoTextEffect-Alignment	Устанавливает/возвращает выравнивание объекта WordArt
FontBold	MsoTriState	Устанавливает/возвращает использование полужирного шрифта
FontItalic	MsoTriState	Устанавливает/возвращает использование курсива
FontName	String	Устанавливает/возвращает шрифт, который используется объектом WordArt
FontSize	Single	Устанавливает/возвращает размер шрифта
KernedPairs	MsoTriState	Устанавливает/возвращает кернинг в объекте WordArt
Normalized-Height	MsoTriState	Устанавливает/возвращает необходимость приведения заглавных и прописных букв к одному размеру
PresetShape	MsoPresetText-EffectShape	Устанавливает/возвращает форму объекта WordArt
PresetText-Effect	MsoPreset-TextEffect	Устанавливает/возвращает эффект, связанный с объектом WordArt

Имя	*Тип возвращаемого значения*	*Описание*
RotatedChars	MsoTriState	Устанавливает/возвращает поворот объекта WordArt на 90 градусов
Text	String	Устанавливает/возвращает текст для объекта WordArt
Tracking	Single	Устанавливает/возвращает интервал между символами

Методы объекта TextEffectFormat

Имя	*Тип возвращаемого значения*	*Параметры*	*Описание*
Toggle-VerticalText			Переключает вертикальное и горизонтальное отображение текста

Пример: объект TextEffectFormat

```
Sub FormatTextArt()
   Dim oTEF As TextEffectFormat
   Dim oShp As Shape
   Set oShp = ActiveSheet.Shapes(1)
   If oShp.Type = msoTextEffect Then
      Set oTEF = oShp.TextEffect

      With oTEF
         .FontName = "Times New Roman"
         .FontBold = True
         .PresetTextEffect = msoTextEffect14
         .Text = "Здравствуй, мир!"
      End With
   End If
End Sub
```

Объект TextFrame

В этом объекте хранятся свойства и методы, позволяющие управлять формой текстового фрейма. В качестве возможных родительских объектов могут выступать Shape и ShapeRange.

Общие свойства объекта TextFrame

Определения свойств Application, Creator и Parent приводились в начале приложения.

Свойства объекта TextFrame

Имя	*Тип возвращаемого значения*	*Описание*
AutoMargins	Boolean	Устанавливает/возвращает необходимость автоматического вычисления полей текстового фрейма. Устанавливается в значение False для использования свойств MarginLeft, MarginRight, MarginTop и MarginBottom
AutoSize	Boolean	Устанавливает/возвращает автоматическое изменение размера текстового фрейма в соответствии с текстом внутри фрейма

Имя	*Тип возвращаемого значения*	*Описание*
`HorizontalAlignment`	`XlHAlign`	Устанавливает/возвращает горизонтальное выравнивание для текстового фрейма. Для установки этого свойства используются константы `XlHAlign`
`MarginBottom`	`Single`	Устанавливает/возвращает размер нижнего поля текстового фрейма
`MarginLeft`	`Single`	Устанавливает/возвращает размер левого поля текстового фрейма
`MarginRight`	`Single`	Устанавливает/возвращает размер правого поля текстового фрейма
`MarginTop`	`Single`	Устанавливает/возвращает размер верхнего поля текстового фрейма
`Orientation`	`MsoTextOrientation`	Устанавливает/возвращает ориентацию текста в текстовом фрейме
`ReadingOrder`	`Long`	Устанавливает/возвращает направление чтения текста в текстовом фрейме: слева направо (`xlLTR`), справа налево (`xlRTL`) или в зависимости от контекста (`xlContext`)
`VerticalAlignment`	`XlVAlign`	Устанавливает/возвращает вертикальное выравнивание для текстового фрейма. Для установки этого свойства используются константы `XlVAlign`

Методы объекта TextFrame

Имя	*Тип возвращаемого значения*	*Параметры*	*Описание*
`Characters`	`Characters`	`[Start], [Length]`	Возвращает объект, содержащий все символы из текстового фрейма. Позволяет управлять каждым символом в отдельности и извлекать фрагмент текста из текстового фрейма

Пример: объект TextFrame

```
Sub SetShapeAutoSized()
   Dim oTF As TextFrame
   Dim oShp As Shape
   Set oShp = ActiveSheet.Shapes(1)
   Set oTF = oShp.TextFrame
   oTF.AutoSize = True
End Sub
```

Объект ThreeDFormat

В этом объекте хранятся все параметры трехмерного форматирования для родительского объекта `Shape`. Для доступа к объекту `ThreeDFormat` используется свойство `ThreeD` родительского объекта `Shape`.

Общие свойства объекта ThreeDFormat

Определения свойств `Application`, `Creator` и `Parent` приводились в начале приложения.

Свойства объекта ThreeDFormat

Имя	*Тип возвращаемого значения*	*Описание*
`Depth`	`Single`	Устанавливает/возвращает "глубину" трехмерной фигуры
`Extrusion-Color`	`ColorFormat`	Только чтение. Возвращает объект, поддерживающий управление цветом экструзии
`Extrusion-ColorType`	`MsoExtrusion-ColorType`	Устанавливает/возвращает способ установки цвета экструзии
`Perspective`	`MsoTriState`	Устанавливает/возвращает наличие перспективы в экструзии
`Preset-Extrusion-Direction`	`MsoPreset-Extrusion-Direction`	Только чтение. Возвращает направление экструзии
`Preset-Lighting-Direction`	`MsoPreset-Lighting-Direction`	Устанавливает/возвращает направление источника света
`Preset-Lighting-Softness`	`MsoPreset-Lighting-Softness`	Устанавливает/возвращает мягкость источника света
`Preset-Material`	`MsoPreset-Material`	Устанавливает/возвращает материал поверхности экструзии
`Preset-ThreeDFormat`	`MsoPreset-ThreeD`	Только чтение. Возвращает предопределенный формат экструзии
`RotationX`	`Single`	Устанавливает/возвращает поворот экструзии (в градусах)
`RotationY`	`Single`	Устанавливает/возвращает поворот экструзии (в градусах)
`Visible`	`MsoTriState`	Устанавливает/возвращает необходимость отображения трехмерной фигуры

Методы объекта ThreeDFormat

Имя	*Тип возвращаемого значения*	*Параметры*	*Описание*
`Increment-RotationX`		`Increment As Single`	Меняет значение свойства `RotationX`
`Increment-RotationY`		`Increment As Single`	Меняет значение свойства `RotationY`
`Reset-Rotation`			Сбрасывает свойства `RotationX` и `RotationY` в значение 0
`Set-Extrusion-Direction`		`PresetExtrusionDirection As MsoPreset-ExtrusionDirection`	Меняет направление экструзии
`SetThreeD-Format`		`PresetThreeDFormat As MsoPresetThreeD-Format`	Устанавливает предопределенный формат экструзии

Пример: объект ThreeDFormat

```
Sub SetShape3D()
   Dim o3DF As ThreeDFormat
   Dim oShp As Shape
   Set oShp = ActiveSheet.Shapes(1)
```

```
    Set o3DF = oShp.ThreeD
    With o3DF
        .Depth = 10
        .SetExtrusionDirection msoExtrusionBottomRight
    End With
End Sub
```

Объект TickLabels

Объект содержит параметры форматирования, связанные с метками делений на оси диаграммы. Родительским для объекта TickLabels является объект Axis.

Общие свойства объекта TickLabels

Определения свойств Application, Creator и Parent приводились в начале приложения.

Свойства объекта TickLabels

Имя	*Тип возвращаемого значения*	*Описание*
Alignment	Long	Устанавливает/возвращает выравнивание меток делений. Для установки этого свойства используются константы XlHAlign
AutoScale-Font	Variant	Устанавливает/возвращает необходимость автоматического изменения размера шрифта при изменении размера родительской диаграммы
Depth	Long	Только чтение. Возвращает количество уровней меток делений категорий на диаграмме
Font	Font	Только чтение. Возвращает объект с параметрами шрифта текста меток делений
Name	String	Только чтение. Возвращает имя объекта TickLabels
NumberFormat	String	Устанавливает/возвращает формат чисел для меток делений, содержащих числа или даты
Number-FormatLinked	Boolean	Устанавливает/возвращает необходимость использования метками делений того же формата чисел, который применяется в ячейках с исходными данными диаграммы
Number-FormatLocal	Variant	Устанавливает/возвращает формат чисел для меток делений, содержащих числа и даты. Формат возвращается на языке пользовательского интерфейса
Offset	Long	Устанавливает/возвращает расстояние между уровнями меток в сравнении с размером шрифта метки оси
Orientation	XlTickLabel-Orientation	Устанавливает/возвращает угол расположения текста в метках делений. Значение может указываться в градусах (от -90 до 90). Кроме этого, можно воспользоваться константами XlTickLabelOrientation
ReadingOrder	Long	Устанавливает/возвращает направление чтения текста (слева направо или справа налево). Свойство используется с соответствующими языками

Методы объекта TickLabels

Имя	*Тип возвращаемого значения*	*Параметры*	*Описание*
Delete	Variant		Удаляет метки делений из меток оси
Select	Variant		Выделяет метки делений на диаграмме

Пример: объект TickLabels

```
Sub FormatTickLabels()
   Dim oTL As TickLabels
   Set oTL = Charts(1).Axes(xlValue).TickLabels
   With oTL
      .NumberFormat = "#,##0"
      .Font.Size = 12
   End With
End Sub
```

Объект TreeviewControl

Данный объект обеспечивает управление иерархическим выделением в пределах поля куба. Обычно этот объект используется механизмом записи макросов и не применяется при самостоятельном создании кода VBA. Родительским для объекта TreeviewControl является объект CubeField.

Общие свойства объекта TreeviewControl

Определения свойств Application, Creator и Parent приводились в начале приложения.

Свойства TreeviewControl

Имя	*Тип возвращаемого значения*	*Описание*
Drilled	Variant	Устанавливает/возвращает строковый массив, описывающий состояние (развернут/свернут) членов родительского поля куба
Hidden	Variant	Устанавливает/возвращает необходимость сокрытия члена поля куба

Объект Trendline и коллекция Trendlines

В коллекции Trendlines хранятся линии трендов в пределах диаграммы. Каждый объект Trendline соответствует одной линии тренда для определенной последовательности. Объекты Trendline используются для графического представления трендов и позволяют прогнозировать значения в будущем. Родительским объектом коллекции Trendlines является объект Series.

Кроме типичных атрибутов, коллекция Trendlines предоставляет одно свойство и один метод. Свойство Count возвращает количество объектов Trendline в коллекции. Метод Add добавляет линию тренда на текущую диаграмму. Метод Add принимает параметры Type, Order, Period, Forward, Backward, Intercept, DisplayEquation, DispayRSquared и Name. Дополнительная информация об этих свойствах приводится в разделе “Свойства объекта Trendline”.

Общие свойства объекта Trendline

Определения свойств Application, Creator и Parent приводились в начале приложения.

Свойства объекта Trendline

Имя	*Тип возвращаемого значения*	*Описание*
Backward	Long	Устанавливает/возвращает количество периодов, для которых рассчитывается линия тренда
Border	Border	Только чтение. Возвращает свойства границы вокруг линии тренда
DataLabel	DataLabel	Только чтение. Возвращает объект, поддерживающий управление метками данных линии тренда
Display-Equation	Boolean	Устанавливает/возвращает необходимость вывода в пределах диаграммы уравнения, на основе которого создается линия тренда
DisplayR-Squared	Boolean	Устанавливает/возвращает необходимость отображения квадратичного значения линии тренда в пределах диаграммы
Forward	Long	Устанавливает/возвращает количество периодов в будущем, для которых вычисляется линия тренда
Index	Long	Только чтение. Возвращает положение текущего объекта в коллекции
Intercept	Double	Устанавливает/возвращает точку, в которой линия тренда пересекает ось значений (y)
Intercept-IsAuto	Boolean	Устанавливает/возвращает необходимость автоматического вычисления точки пересечения линии тренда с осью значений с помощью регрессии
Name	String	Устанавливает/возвращает имя объекта Trendline
NameIsAuto	Boolean	Устанавливает/возвращает необходимость автоматического выбора имени для линии тренда
Order	Long	Устанавливает/возвращает порядок полиномиальной линии тренда. При этом свойство Type должно быть установлено в значение xlPolynomial
Period	Long	Устанавливает/возвращает период линии тренда для средних значений
Type	XlTrendline-Type	Устанавливает/возвращает тип линии тренда (например, xlExponential, xlLinear и т.д.)

Методы объекта Trendline

Имя	*Тип возвращаемого значения*	*Параметры*	*Описание*
ClearFormats	Variant		Очищает форматирование линий тренда
Delete	Variant		Удаляет линию тренда
Select	Variant		Выделяет линию тренда на диаграмме

Пример: объект Trendline и коллекция Trendlines

```
Sub AddTrendLine()
   Dim oSer As Series
   Dim oTL As Trendline
   Set oSer = Charts(1).SeriesCollection(1)
   Set oTL = oSer.Trendlines.Add(xlLinear)
   With oTL
      .DisplayEquation = True
      .DisplayRSquared = True
   End With
End Sub
```

Объект UpBars

Этот объект содержит параметры форматирования для полос повышения на диаграмме. Родительским для объекта UpBars является объект ChartGroup. Для проверки существования этого объекта можно воспользоваться свойством HasUpDownBars объекта ChartGroup.

Общие свойства объекта UpBars

Определения свойств Application, Creator и Parent приводились в начале приложения.

Свойства объекта UpBars

Имя	*Тип возвращаемого значения*	*Описание*
Border	Border	Только чтение. Возвращает свойства границы вокруг полос повышения
Fill	ChartFillFormat	Только чтение. Возвращает объект, содержащий параметры форматирования заливки полос повышения
Interior	Interior	Только чтение. Возвращает объект, содержащий параметры форматирования внутренней области полос повышения (например, цвет внутренней области)
Name	String	Только чтение. Возвращает имя полос повышения

Методы объекта UpBars

Имя	*Тип возвращаемого значения*	*Параметры*	*Описание*
Delete	Variant		Удаляет полосы повышения
Select	Variant		Выделяет полосы повышения на диаграмме

Пример: объект UpBars

```
Sub AddAndFormatUpBars()
   Dim oUpBars As UpBars
   ' Добавить полосы повышения/понижения на диаграмму
   Charts(1).ChartGroups(1).HasUpDownBars = True
   ' Получить коллекцию объектов UpBars
   Set oUpBars = Charts(1).ChartGroups(1).UpBars
   ' Отформатировать полосы повышения
   With oUpBars
      .Interior.ColorIndex = 3
```

```
        .Interior.Pattern = xlSolid
    End With
End Sub
```

Коллекция UsedObjects

Эта коллекция предоставляет доступ ко всем объектам, используемым в данный момент во всех открытых книгах. В качестве таких объектов могут выступать листы, листы диаграмм и сами книги, а также любые элементы управления ActiveX, размещенные в книгах. Доступ к объекту предоставляется через объект `Application`.

Обратите внимание, что кроме типичных свойств, коллекция `UsedObjects` предоставляет свойства `Item` и `Count`.

Пример: коллекция UsedObjects

Следующая подпрограмма перечисляет все родительские объекты коллекции `UsedObjects`:

```
Sub CountUsedObjects()
    Dim lCount As Long
    Dim oUsedObjs As UsedObjects
    ' Отключить обновление экрана
    Application.ScreenUpdating = False

    ' Сохранить коллекцию используемых объектов
    Set oUsedObjs = Application.UsedObjects

    ' Очистить старый список
    wksUsedObjects.UsedRange.Offset(1, 0).Resize(, 1).ClearContents
    ' Перебрать список родительских объектов для всех объектов
    ' Не получается извлечь имя/заголовок/... самого объекта
    For lCount = 1 To oUsedObjs.Count
        wksUsedObjects.Range("ListStart").Cells(lCount, 1) =
➥ oUsedObjs.Item(lCount).Parent.Name
    Next lCount
End Sub
```

Коллекция UserAccess

Описывает одного пользователя из группы пользователей, которым разрешен доступ к диапазону, на который указывает объект `AllowEditRange`. На пользователя можно ссылаться через свойство `Item` объекта `UserAccessList`. После получения ссылки свойства этого объекта позволяют поменять параметры пользователя.

Свойства коллекции UserAccess

Имя	*Тип возвращаемого значения*	*Описание*
`AllowEdit`	`Boolean`	Устанавливает/возвращает разрешение на доступ к указанному диапазону на защищенном листе
`Name`	`String`	Только чтение. Возвращает имя объекта `UserAccess`

Методы коллекции UserAccess

Имя	Тип возвращаемого значения	Параметры	Описание
`Delete`			Удаляет объект

Коллекция UserAccessList

Описывает список пользователей, которым предоставлен доступ к защищенному диапазону листа. Доступ к этому объекту предоставляется через объект `AllowEditRange` (после его создания). Для добавления пользователя в список воспользуйтесь методом `Add`. Параметры метода `Add` позволяют указать необходимость получения пароля при предоставлении доступа.

Обратите внимание, что пароль устанавливается с помощью метода `ChangePassword` объекта `AllowEditRange`. Это значит, что все пользователи из списка объекта `AllowEditRange` будут вводить один и тот же пароль. Обратите внимание, что эта коллекция предоставляет только свойства `Count` и `Item`.

Методы коллекции UserAccessList

Имя	Тип возвращаемого значения	Параметры	Описание
`Add`	`UserAccess`	`Name As String, AllowEdit As Boolean`	Добавляет в коллекцию список доступа пользователей. Список передается в качестве значения параметра `Name`. Если параметр `AllowEdit` установлен в значение `True`, пользователи из списка доступа получают право редактирования диапазонов на защищенном листе
`DeleteAll`			Удаляет всех пользователей из списка управления доступом к защищенному диапазону на листе

Пример: коллекция UserAccessList

Следующая подпрограмма перебирает все объекты `AllowEditRange` на указанном листе и удаляет всех пользователей, кроме пользователей, имеющих доступ к диапазону `pcNetSales`.

```
Sub DeleteAllUsers()

    Dim oAllowRange As AllowEditRange

    ' Перебрать все объекты AllowEditRange на указанном листе
    For Each oAllowRange In wksAllowEditRange.Protection.AllowEditRanges
        ' Удаляет все элементы из всех объектов AllowEditRanges
        ' кроме диапазона, в котором объект AllowEditRanges имеет
        ' заголовок pcNetSales
        If oAllowRange.Title <> "pcNetSales" Then
            oAllowRange.Users.DeleteAll
        End If
    Next oAllowRange

End Sub
```

Объект Validation

Объект Validation содержит свойства и методы для проверки действительности введенных в диапазон значений. Родительским для объекта Validation является объект Range.

Общие свойства объекта Validation

Определения свойств Application, Creator и Parent приводились в начале приложения.

Свойства объекта Validation

Имя	*Тип возвращаемого значения*	*Описание*
AlertStyle	Long	Только чтение. Возвращает тип предупреждения, которое выдается пользователю при вводе некорректных данных. Для установки этого свойства используются константы XlDVAlertStyle
ErrorMessage	String	Устанавливает/возвращает сообщение об ошибке, которое выдается при неудачной проверке действительности введенных данных
ErrorTitle	String	Устанавливает/возвращает заголовок диалогового окна с сообщением об ошибке
Formula1	String	Только чтение. Возвращает значение, ссылку на ячейку или формулу, которые используются для проверки действительности данных
Formula2	String	Только чтение. Возвращает вторую часть значения, ссылки на ячейку или формулу, которые используются для проверки действительности данных. Свойство Operator должно быть установлено в значение xlBetween или xlNotBetween
IgnoreBlank	Boolean	Устанавливает/возвращает автоматически положительный результат проверки для пустых ячеек
IMEMode	Long	Устанавливает/возвращает описание правил ввода для японского языка. Для установки этого свойства используются константы XlIMEMode
InCell-Dropdown	Boolean	Устанавливает/возвращает отображения раскрывающегося списка с действительными значениями в родительском диапазоне. Используется, если свойство Type установлено в значение xlValidateList
InputMessage	String	Устанавливает/возвращает сообщение, которое выдается при запросе действительных данных у пользователя
Operator	Long	Только чтение. Возвращает оператор, описывающий применение свойств Formula1 и Formula2 для проверки действительности введенных данных. Для установки этого свойства используются константы XlFormatCondition-Operator
ShowError	Boolean	Устанавливает/возвращает отображение сообщения об ошибке при вводе некорректных данных в родительский диапазон

Имя	*Тип возвращаемого значения*	*Описание*
ShowInput	Boolean	Устанавливает/возвращает отображение запроса на ввод, когда пользователь выбирает одну из ячеек в родительском диапазоне
Type	Long	Только чтение. Возвращает тип проверки действительности данных для родительского диапазона. Для установки этого свойства используются константы XlDVType (например, xlValidateDecimal, xlValidateTime)
Value	Boolean	Только чтение. Возвращает результаты проверки для диапазона

Методы объекта Validation

Имя	*Тип возвращаемого значения*	*Параметры*	*Описание*
Add		Type As XlDVType, [AlertStyle], [Operator], [Formula1], [Formula2]	Добавляет проверку действительности данных в родительский диапазон. При этом необходимо указать тип проверки (параметр Type). Тип предупреждения для проверки (параметр AlertStyle) устанавливается с помощью констант XlDVAlertStyle. Для установки значения параметра Operator используются константы XlFormatConditionOperator. Параметры Formula1 и Formula2 описывают формулу проверки действительности данных
Delete			Удаляет объект Validation из диапазона
Modify		[Type], [AlertStyle], [Operator], [Formula1], [Formula2]	Модифицирует свойства объекта Validation. Описание параметров приводится в разделе, посвященном свойствам объекта Validation

Пример: объект Validation

```
Sub AddValidation()
   Dim oValid As Validation
   Set oValid = Selection.Validation
   With oValid
       .Delete
       .Add Type:=xlValidateWholeNumber, AlertStyle:=xlValidAlertStop, _
          Operator:=xlBetween, Formula1:="10", Formula2:="20"
       .ShowInput = False
       .ShowError = True
       .ErrorTitle = "Error"
       .ErrorMessage = "Число должно находится в пределах от 10 до 20"
   End With
End Sub
```

Объект VPageBreak и коллекция VPageBreaks

В коллекции VPageBreaks хранятся все вертикальные разрывы страницы печатаемой области родительского объекта. Каждый объект VPageBreak соответствует одному вертикальному разрыву страницы в печатаемой области родительского объекта. В качестве возможных родительских объектов для коллекции VPageBreaks могут выступать объекты Worksheet и Chart.

Кроме типичных атрибутов, коллекция VPageBreaks предоставляет одно свойство и один метод. Свойство Count возвращает количество объектов VPageBreak в коллекции. Метод Add позволяет добавить в коллекцию объект VPageBreak (при этом на странице появляется вертикальный разрыв). Метод Add принимает параметр Before, который описывает положение диапазона справа от вертикального разрыва страницы.

Общие свойства объекта VPageBreak

Определения свойств Application, Creator и Parent приводились в начале приложения.

Свойства объекта VPageBreak

Имя	*Тип возвращаемого значения*	*Описание*
Extent	XlPageBreak-Extent	Только чтение. Возвращает протяженность разрыва страницы на весь экран или в пределах области печати
Location	Range	Устанавливает/возвращает ячейку, в которой расположен вертикальный разрыв страницы. Левый край ячейки совпадает с вертикальным разрывом страницы
Type	XlPageBreak	Устанавливает/возвращает тип разрыва страницы (автоматический или добавленный вручную)

Методы объекта VPageBreak

Имя	*Тип возвращаемого значения*	*Параметры*	*Описание*
Delete			Удаляет разрыв страницы
DragOff		Direction As XlDirection, RegionIndex As Long	Перемещает разрыв страницы за пределы области печати. Параметр Direction определяет направление перемещения разрыва страницы. Параметр RegionIndex определяет диапазон печати, в который перемещается разрыв страницы

Пример: объект VPageBreak и коллекция VPageBreaks

```
Sub AddVPageBreaks()
   Dim oCell As Range
   ' Перебрать все ячейки в первом столбце листа
   For Each oCell In ActiveSheet.UsedRange.Rows(1).Cells
      ' Если размер шрифта установлен в занчение 16,
      ' добавить разрыв страницы слева от ячейки
      If oCell.Font.Size = 16 Then
         ActiveSheet.VPageBreaks.Add oCell
      End If
   Next
End Sub
```

Объект Walls

В этом объекте хранятся параметры форматирования для всех вертикальных плоскостей трехмерной диаграммы. Доступ к отдельным вертикальным плоскостям трехмерной диаграммы невозможен. Родительским для объекта `Walls` является объект `Chart`.

Общие свойства объекта Walls

Определения свойств `Application`, `Creator` и `Parent` приводились в начале приложения.

Свойства объекта Walls

Имя	*Тип возвращаемого значения*	*Описание*
`Border`	`Border`	Только чтение. Возвращает свойства границы вокруг плоскостей трехмерной диаграммы
`Fill`	`ChartFillFormat`	Только чтение. Возвращает объект, содержащий параметры форматирования заливки вертикальных плоскостей трехмерной диаграммы
`Interior`	`Interior`	Только чтение. Возвращает объект, содержащий параметры форматирования внутренней области вертикальных плоскостей трехмерной диаграммы (например, цвета внутренней области)
`Name`	`String`	Только чтение. Возвращает имя объекта `Walls`
`PictureType`	`Variant`	Устанавливает/возвращает способ отображения рисунка, связанного с вертикальными плоскостями трехмерной диаграммы (например, растягивание, в виде мозаики). Для установки этого свойства используются константы `XlPictureType`
`PictureUnit`	`Variant`	Устанавливает/возвращает количество единиц измерения, которое соответствует рисунку, когда свойство `PictureType` установлено в значение `xlScale`

Методы объекта Walls

Имя	*Тип возвращаемого значения*	*Параметры*	*Описание*
`ClearFormats`	`Variant`		Очищает форматирование объектов `Walls`
`Delete`			Удаляет объект `Walls`
`Select`	`Variant`		Выделяет вертикальные плоскости на родительской диаграмме

Пример: объект Walls

```
Sub FormatWalls()
   Dim oWall As Walls
   Set oWall = Charts(1).Walls
   With oWall
      .Fill.PresetTextured msoTextureCork
      .Fill.Visible = True
   End With
End Sub
```

Объект Watch и коллекция Watches

Объект `Watch` соответствует одному контрольному значению в окне Контрольные значения (Watch). Для доступа к этому окну выберите команду меню Вид⇨Панели инструментов⇨Окно контрольного значения (View⇨Toolbars⇨Watch). Каждое контрольное значение может описывать ячейку или диапазон ячеек, за которыми нужно следить в процессе изменения данных на листе. Объект `Watch` напоминает контрольные значения, которые создаются в редакторе VBE. Контрольные значения позволяют следить за изменениями в ячейках или диапазонах ячеек в процессе изменения данных на листе.

В коллекции `Watches` хранятся все объекты `Watch` в пределах приложения.

Методы коллекции Watches

Имя	*Тип возвращаемого значения*	*Параметры*	*Описание*
`Add`	`Watch`	`Source As Variant`	Добавляет диапазон, который отслеживается при пересчете содержимого листа
`Delete`			Удаляет объект из коллекции

Общие свойства объекта Watch

Определения свойств `Application`, `Creator` и `Parent` приводились в начале приложения.

Свойства объекта Watch

Имя	*Тип возвращаемого значения*	*Описание*
`Source`	`Variant`	Только чтение. Возвращает уникальное имя, идентифицирующее элементы, свойство `SourceType` которых установлено в значения `xlSourceRange`, `xlSourceChart`, `xlSourcePrintArea`, `xlSourceAutoFilter`, `xlSourcePivotTable` или `xlSourceQuery`

Методы объекта Watch

Имя	*Тип возвращаемого значения*	*Параметры*	*Описание*
`Delete`			Удаляет объект из коллекции

Пример: объект Watch

Следующая подпрограмма запрашивает диапазон у пользователя. После этого она перебирает ячейки в диапазоне и добавляет их в Окно контрольного значения (Watch Window). После чего на экран выводится Окно контрольного значения (Watch Window).

```
Sub AddWatches()
    Dim oWatch As Watch
    Dim rng As Range
    Dim rngWatches As Range

    ' Запросить у пользователя диапазон
    ' Подавить сообщение об ошибке при щелчке на кнопке Отмена
    On Error Resume Next
        Set rngWatches = Application.InputBox(_
```

```
        "Выберите ячейку или диапазон ячеек для контрольного значения",
↳ "Добавление контрольного значения", 8)
    On Error GoTo 0

    ' Если пользователь выбрал диапазон
    If Not rngWatches Is Nothing Then
        ' Перебрать каждую ячейку и добавить ее в список контрольных
        ' значений
        For Each rng In rngWatches
            Application.Watches.Add rng
        Next rng
    End If
    ' Вывести окно контрольных значений в соответствии с запросом
    ' пользователя
    Application.CommandBars("Watch Window").Visible = (Not rngWatches Is
↳ Nothing)

End Sub
```

Объект WebOptions

Объект `WebOptions` содержит атрибуты открытия и сохранения Web-документов. Родительским для объекта `WebOptions` является объект `Workbook`. Установленные свойства объекта `WebOptions` переопределяют параметры объекта `DefaultWebOptions`.

Общие свойства объекта WebOptions

Определения свойств `Application`, `Creator` и `Parent` приводились в начале приложения.

Свойства объекта WebOptions

Имя	*Тип возвращаемого значения*	*Описание*
`AllowPNG`	`Boolean`	Устанавливает/возвращает использование формата Portable Network Graphics Format (PNG) в качестве выходного формата изображений. Этот формат обеспечивает хорошую степень сжатия без потерь и широкую совместимость
`Download-Components`	`Boolean`	Устанавливает/возвращает необходимость загрузки компонентов Office на компьютер пользователя при просмотре файлов Excel в окне Web-обозревателя
`Encoding`	`MsoEncoding`	Устанавливает/возвращает тип кодовой страницы или набора символов, с использованием которых будет сохраняться документ
`Folder-Suffix`	`String`	Только чтение. Возвращает суффикс, который используется для создания имени каталога поддержки при сохранении документа Excel в виде Web-страницы. Значение этого свойства зависит от языка пользовательского интерфейса
`LocationOf-Components`	`String`	Устанавливает/возвращает адрес URL или путь, по которому можно получить доступ к Web-компонентам Office, необходимым для просмотра документов в окне Web-обозревателя
`OrganizeIn-Folder`	`Boolean`	Устанавливает/возвращает необходимость сохранения файлов поддержки в отдельном каталоге

Имя	*Тип возвращаемого значения*	*Описание*
PixelsPer-Inch	Long	Устанавливает/возвращает плотность изображений и ячеек таблицы при просмотре в окне Web-обозревателя
RelyOnCSS	Boolean	Устанавливает/возвращает использование каскадных листов стилей (CSS) для форматирования шрифта
RelyOnVML	Boolean	Устанавливает/возвращает отказ от сохранения файлов изображений при сохранении документа с векторными рисунками. Для создания изображений на лету используется язык Vector Markup Language. Язык VML основан на языке разметки XML и предназначен для описания векторных изображений на Web-страницах
ScreenSIze	MsoScreenSize	Устанавливает/возвращает размер экрана на интересующем мониторе
Target-Browser	MsoTarget-Browser	Устанавливает/возвращает версию обозревателя
UseLong-FileNames	Boolean	Устанавливает/возвращает необходимость обновления ссылок при каждом сохранении документа

Методы объекта WebOptions

Имя	*Тип возвращаемого значения*	*Параметры*	*Описание*
UseDefault-FolderSuffix			Заставляет Excel использовать принятую по умолчанию схему именования для создания каталогов поддержки

Пример: объект WebOptions

```
Sub SetWebOptions()
   Dim oWO As WebOptions
   Set oWO = ActiveWorkbook.WebOptions
   With oWO
      .ScreenSize = msoScreenSize800x600
      .RelyOnCSS = True
      .UseDefaultFolderSuffix
   End With
End Sub
```

Объект Window и коллекция Windows

В коллекции Windows хранится список окон, которые используются в Excel или в пределах книги. Каждый объект Window соответствует одному окну Excel, содержащему полосы прокрутки и сетку направляющих. Родительским объектом коллекции Windows может быть объект `Application` или объект `Workbook`.

Кроме типичных атрибутов, коллекция Windows предоставляет свойство `Count` и метод `Arrange`. Свойство `Count` возвращает количество объектов `Window` в коллекции. Метод `Arrange` выстраивает окна в коллекции в соответствии со значением параметра `ArrangeStyle`. Для установки параметра `ArrangeStyle` используются константы `XlArrangeStyle`. Установите параметр `ActiveWorkbook` в значение `True` для выстраивания только тех окон, которые связаны с открытой книгой. А параметр `SyncHorizontal` или параметр `SyncVertical` установите в значение `True` для синхронной горизонтальной или вертикальной прокрутки окон.

Общие свойства объекта Window

Определения свойств Application, Creator и Parent приводились в начале приложения.

Свойства объекта Window

Имя	*Тип возвращаемого значения*	*Описание*
ActiveCell	Range	Только чтение. Возвращает ячейку в окне, в котором расположен курсор
ActiveChart	Chart	Только чтение. Возвращает текущую выделенную диаграмму в окне. Если не выделено ни одной диаграммы, свойство ничего не возвращает
ActivePane	Pane	Только чтение. Возвращает активную панель в пределах окна
ActiveSheet		Только чтение. Возвращает активный лист в пределах окна
Caption	Variant	Устанавливает/возвращает заголовок окна
Display-Formulas	Boolean	Устанавливает/возвращает необходимость отображения формул в окне. Не используется с листами диаграмм
Display-Gridlines	Boolean	Устанавливает/возвращает необходимость отображения направляющих на листе
Display-Headings	Boolean	Устанавливает/возвращает необходимость отображения заголовков столбцов. Не используется с листами диаграмм
Display-Horizontal-ScrollBar	Boolean	Устанавливает/возвращает необходимость отображения горизонтальной полосы прокрутки
Display-Outline	Boolean	Устанавливает/возвращает необходимость отображения символов структуры
Display-RightToLeft	Boolean	Устанавливает/возвращает необходимость вывода содержимого окна справа налево. Используется только с языками, поддерживающими направление чтения справа налево
Display-Vertical-ScrollBar	Boolean	Устанавливает/возвращает необходимость отображения вертикальной полосы прокрутки
Display-WorkbookTabs	Boolean	Устанавливает/возвращает необходимость отображения вкладок книг
DisplayZeros	Boolean	Устанавливает/возвращает необходимость вывода нулевых значений. Не используется с листами диаграмм
EnableResize	Boolean	Устанавливает/возвращает возможность изменения размера окна через пользовательский интерфейс
FreezePanes	Boolean	Устанавливает/возвращает фиксацию разделенных панелей. Не используется с листами диаграмм
Gridline-Color	Long	Устанавливает/возвращает цвет направляющих. Для создания значения цвета используется функция RGB
Gridline-ColorIndex	XlColorIndex	Устанавливает/возвращает цвет направляющих. Для установки свойства используются константы XlColorIndex или номер цвета в текущей цветовой палитре
Height	Double	Устанавливает/возвращает высоту окна

Имя	*Тип возвращаемого значения*	*Описание*
Index	Long	Только чтение. Возвращает положение текущего объекта в коллекции
Left	Double	Устанавливает/возвращает расстояние от левого края клиентской области до левого края окна
OnWindow	String	Устанавливает/возвращает имя процедуры, которая запускается при активизации окна
Panes	Panes	Только чтение. Возвращает панели в пределах окна
Range-Selection	Range	Только чтение. Возвращает выделенный диапазон ячеек или объект в пределах окна
Scroll-Column	Long	Устанавливает/возвращает номер самого левого видимого столбца в пределах окна
ScrollRow	Long	Устанавливает/возвращает номер самой верхней видимой строки в пределах окна
Selected-Sheets	Sheets	Только чтение. Возвращает выделенные листы в пределах окна
Selection	Object	Только чтение. Возвращает выделенный в пределах окна объект
Split	Boolean	Устанавливает/возвращает разбиение окна на панели
SplitColumn	Long	Устанавливает/возвращает номер столбца, разделяющей окно на панели
Split-Horizontal	Double	Устанавливает/возвращает положение горизонтального разбиения окна (в пунктах)
SplitRow	Long	Устанавливает/возвращает номер строки, по которой окно было разбито на панели
Split-Vertical	Double	Устанавливает/возвращает положение вертикального разбиения окна (в пунктах)
Sync-Scrolling-SideBySide	Boolean	Устанавливает/возвращает одновременную прокрутку документов при сравнении
TabRatio	Double	Устанавливает/возвращает размер вкладки книги как отношение ширины области вкладок к ширине горизонтальной полосы прокрутки окна
Top	Double	Устанавливает/возвращает расстояние от верхнего края клиентской области до верхнего края окна
Type	XlWindowType	Только чтение. Возвращает тип окна
UsableHeight	Double	Только чтение. Возвращает максимальную высоту окна
UsableWidth	Double	Только чтение. Возвращает максимальную ширину окна
View	XlWindowView	Устанавливает/возвращает представление окна (например, `xlNormalView`, `xlPageBreakPreview`)
Visible	Boolean	Устанавливает/возвращает видимость окна
VisibleRange	Range	Только чтение. Возвращает видимый в текущем окне диапазон ячеек
Width	Double	Устанавливает/возвращает ширину окна
WindowNumber	Long	Только чтение. Возвращает номер окна. Обычно этот номер используется при повторном открытии одной и той же книги (например, `MyBook.xls:1` и `MyBook.xls:2`)

Имя	*Тип возвращаемого значения*	*Описание*
`WindowState`	`XlWindowState`	Устанавливает/возвращает состояние окна (свернуто, развернуто или нормальное)
`Zoom`	`Variant`	Устанавливает/возвращает масштаб содержимого окна

Методы объекта Window

Имя	*Тип возвращаемого значения*	*Параметры*	*Описание*
`Activate`	`Variant`		Активизирует окно
`ActivateNext`	`Variant`		Активизирует следующее окно в соответствии с порядком вывода на экран
`Activate-Previous`	`Variant`		Активизирует предыдущее окно в соответствии с порядком вывода на экран
`BreakSide-BySide`	`Boolean`		Отключает режим отображения бок о бок. Возвращаемое значение описывает результат операции
`Close`	`Boolean`	`[SaveChanges], [Filename], [RouteWorkbook]`	Закрывает окно. Для автоматического сохранения изменений свойство `SaveChanges` устанавливается в значение `True`. Если параметр `SaveChanges` установлен в значение `False`, все изменения будут потеряны. Параметр `Filename` используется для указания имени файла, в котором сохраняется окно. Параметр `RouteWorkbook` позволяет автоматически передать книгу следующему получателю
`CompareSide-BySideWith`	`Boolean`		Открывает два окна в режиме отображения бок о бок
`LargeScroll`	`Variant`	`[Down], [Up], [ToRight], [ToLeft]`	Прокручивает документ на один экран в соответствующем направлении (направление указывается с помощью параметров)
`NewWindow`	`Window`		Создает и возвращает новое окно
`PointsTo-ScreenPix-elsX`	`Long`	`Points As Long`	Выполняет преобразование горизонтальной координаты документа из параметра `Points` в экранные координаты (в пикселях)
`PointsTo-ScreenPix-elsY`	`Long`	`Points As Long`	Выполняет преобразование вертикальной координаты документа из параметра `Points` в экранные координаты (в пикселях)

Имя	*Тип возвращаемого значения*	*Описание*	
PrintOut	Variant	[From], [To], [Copies], [Preview], [ActivePrinter], [PrintToFile], [Collate], [PrToFileName]	Распечатывает документ в окне. Принтер, количество копий, порядок сортировки и необходимость предварительного просмотра определяются значениями параметров. Кроме этого, печать листа можно перенаправить в файл. Для этого используются параметры PrintToFile и PrToFileName. Параметры From и To определяют диапазон печатаемых страниц
PrintPreview	Variant	[EnableChanges]	Выводит текущую книгу в режиме предварительного просмотра. Для отключения кнопок Поля и Параметры страницы параметр EnableChanges устанавливается в значение False. При этом пользователь не сможет модифицировать параметры страницы в режиме предварительного просмотра
RangeFromPoint	Object	x As Long, y As Long	Возвращает фигуру или диапазон, расположенные в точке с координатами x и y. Если по указанным координатам объект отсутствует, метод ничего не возвращает
ResetPositionsSideBySide			Сбрасывает положение двух окон, расположенных бок о бок
ScrollIntoView		Left As Long, Top As Long, Width As Long, Height As Long, [Start]	Прокручивает окно в положение, определяемое параметрами Left, Top, Width и Height. При этом параметры описывают верхний левый угол окна (Start = True) или нижний правый угол окна (Start = False). Значения параметров Left, Top, Width и Height указываются в пунктах
ScrollWorkbookTabs	Variant	[Sheets], [Position]	Прокручивает список вкладок на указанное количество листов (параметр Sheets) или до указанного листа (параметр Position)
SmallScroll	Variant	[Down], [Up], [ToRight], [ToLeft]	Прокручивает документ на одну строку в указанном направлении

Пример: объект Window и коллекция Windows

```
Sub MinimiseAllWindows()
   Dim oWin As Window
   For Each oWin In Windows
      oWin.WindowState = xlMinimized
   Next
End Sub
```

Объект Workbook и коллекция Workbooks

В коллекции Workbooks хранится список всех открытых книг. Объект Workbook соответствует одной книге. Родительским для объекта Workbook является объект Application.

Свойства коллекции Workbooks

Имя	*Тип возвращаемого значения*	*Описание*
Count	Long	Только чтение. Возвращает количество объектов Workbook в коллекции
DisplayInk-Comments	Boolean	Установка этого свойства в значение True приведет к выводу типографских комментариев
Document-Library-Versions		Возвращает коллекцию всех версий общих книг, в которых включено отслеживание версий
Inactive-ListBorder-Visible	Boolean	Устанавливает видимость границы элемента управления списка при деактивизации элемента управления
Permission		Возвращает объект Permission, описывающий параметры доступа
Shared-Workspace		Возвращает ссылку на рабочее пространство, в котором был создан документ
Smart-Document		Возвращает все параметры смарт-документа
Sync		Предоставляет доступ ко всем методам и свойствам документа, входящего в рабочее пространство
XmlMaps		Предоставляет доступ ко всем отображениям схем, которые были добавлены в книгу
Xml-Namespaces		Возвращает все пространства имен XML, которые были добавлены в книгу

Методы коллекции Workbooks

Имя	*Тип возвращаемого значения*	*Параметры*	*Описание*
Add	Workbook	[Template]	Добавляет новую книгу в коллекцию. Параметру Template присваивается имя шаблона, на основе которого создается книга. Кроме этого, для определения имени книги используются константы XlWBATemplate

Имя	*Тип возвращаемого значения*	*Параметры*	*Описание*
CanCheckOut	Boolean	Filename As String	Возвращает возможность получения указанной книги с сервера
Close			Закрывает книгу
Discard-Conflict		Filename As String	Это ключевое слово зарезервировано для использования в будущем
Offline-Conflict	Boolean	Filename As String	Это ключевое слово зарезервировано для использования в будущем
Open	Workbook	Filename As String, [UpdateLinks], [ReadOnly], [Format], [Password], [WriteResPassword], [IgnoreReadOnlyRecommended], [Origin], [Delimiter], [Editable], [Notify], [Converter], [AddToMru], [Local], [CorruptLoad], [OpenConflict Document]	Открывает указанную книгу. Имя книги передается в качестве значения параметра Filename. Открытая книга добавляется в коллекцию. Параметр UpdateLinks позволяет выбрать способ обновления связей, хранящихся в файле. Для открытия книги в режиме только чтения параметр ReadOnly устанавливается в значение True. Если файл требует предоставления пароля перед открытием, пароль можно присвоить параметру Password или WriteResPassword. Для добавления книги в коллекцию недавно открытых файлов параметр AddToMru необходимо установить в значение True. Если открываемый файл содержит текст с разделителями, можно воспользоваться соответствующими параметрами. Параметр Format позволяет определить символ-разделитель. Параметр Origin определяет кодовую страницу открываемого файла. Параметр Delimiter позволяет указать символ-разделитель, если параметру Format присвоено значение 6 (собственный формат)
OpenDatabase	Workbook	Filename As String, [CommandText], [CommandType], [Background Query], [ImportData As]	Возвращает объект Workbook, соответствующий базе данных, на которую указывает параметр Filename. Параметры CommandText и CommandType позволяют определить текст и тип запроса

Имя	*Тип возвращаемого значения*	*Параметры*	*Описание*
OpenText		Filename As String, [Origin], [StartRow], [DataType], [TextQualifier], [Consecutive Delimiter], [Tab], [Semicolon], [Comma], [Space], [Other], [OtherChar], [FieldInfo], [TextVisual Layout], [DecimalSeparator], [ThousandsSeparator], [TrailingMinusNumbers], [Local]	Открывает текстовый файл, имя которого указано в качестве значения параметра Filename. Содержимое файла размещается на листах создаваемой книги. Параметр Origin позволяет указать кодовую страницу для текста в файле (для установки значения параметра используются параметры XlPlatform). Параметр StartRow определяет первую разбираемую строку. Параметр DataType описывает тип файла: с разделителями (xlDelimited) или с полями фиксированного размера (xlFixedWidth). Если параметр ConsecutiveDelimiter установить в значение True, последовательность разделителей будет восприниматься в качестве одного разделителя. Для выбора символа-разделителя параметр Tab, Semicolon, Comma, Space или Other устанавливается в значение True. Для выбора числовых разделителей используются параметры DecimalSeparator и ThousandsSeparator
OpenXML	Workbook	Filename As String, [Stylesheets]	Загружает в Microsoft Excel файл XML. Для определения страницы стилей XSLT с инструкциями обработки используется параметр Stylesheets

Общие свойства объекта Workbook

Определения свойств Application, Creator и Parent приводились в начале приложения.

Свойства объекта Workbook

Имя	*Тип возвращаемого значения*	*Описание*
AcceptLabels-InFormulas	Boolean	Устанавливает/возвращает возможность использования меток в формулах листа
ActiveChart	Chart	Только чтение. Возвращает активную диаграмму в пределах листа

Имя	*Тип возвращаемого значения*	*Описание*
ActiveSheet		Только чтение. Возвращает активный лист (лист диаграммы или лист электронной таблицы) в пределах книги
AutoUpdate-Frequency	Long	Устанавливает/возвращает частоту автоматического обновления общей книги (в минутах)
AutoUpdate-SaveChanges	Boolean	Устанавливает/возвращает видимость изменений в общей книге для других пользователей при автоматическом обновлении
Builtin-Document-Properties	Document-Properties	Только чтение. Возвращает коллекцию, в которой хранятся все встроенные свойства книги. Этот объект предоставляет доступ к таким атрибутам, как заголовок, тема, автор и количество слов в пределах книги
Calculation-Version	Long	Только чтение. Возвращает номер версии приложения Excel, которое использовалось при последнем пересчете листа
Change-History-Duration	Long	Устанавливает/возвращает длительность истории изменений для совместно используемой книги
Charts	Sheets	Только чтение. Возвращает диаграммы, хранящиеся в книге
CodeName	String	Только чтение. Возвращает имя книги, установленное на этапе проектирования в редакторе VBE
Colors	Variant	Параметры: [Index]. Устанавливает/возвращает цвета для цветовой палитры книги. В палитре может присутствовать до 56 цветов
CommandBars	CommandBars	Только чтение. Возвращает объект, предоставляющий доступ к командным панелям в Excel
Conflict-Resolution	XlSave-Conflict-Resolution	Устанавливает/возвращает способ решения конфликтов при обновлении совместно используемой книги (например, xlLocalSessionChanges означает, что изменения локального пользователя принимаются всегда)
Container		Только чтение. Возвращает объект, содержащий книгу
CreateBackup	Boolean	Только чтение. Возвращает необходимость создания файла с резервной копией при сохранении книги
Custom-Document-Properties	Document-Properties	Только чтение. Возвращает коллекцию, в которой хранятся все определенные пользователем свойства книги
CustomViews	CustomViews	Только чтение. Возвращает коллекцию созданных пользователем представлений книги
Date1904	Boolean	Устанавливает/возвращает необходимость использования в книге системы дат 1904
Display-Drawing-Objects	xlDisplay-Drawing-Objects	Устанавливает/возвращает необходимость вывода геометрических фигур или шаблонов вместо геометрических фигур или сокрытия фигур
EnableAuto-Recover	Boolean	Устанавливает/возвращает периодическое сохранение изменений в файлах всех форматов
Envelope-Visible	Boolean	Устанавливает/возвращает видимость панели инструментов конверта и заголовка создания сообщений электронной почты

Имя	*Тип возвращаемого значения*	*Описание*
`Excel4IntlMacroSheets`	`Sheets`	Только чтение. Возвращает коллекцию интернационализированных листов макросов Excel 4.0
`Excel4MacroSheets`	`Sheets`	Только чтение. Возвращает коллекцию листов макросов Excel 4.0
`FileFormat`	`XlFileFormat`	Только чтение. Возвращает формат файла для книги
`FullName`	`String`	Только чтение. Возвращает путь и имя файла, в котором хранится книга
`FullNameURLEncoded`	`String`	Только чтение. Возвращает имя объекта, включая его путь на диске
`HasPassword`	`Boolean`	Только чтение. Возвращает наличие парольной защиты книги
`HasRoutingSlip`	`Boolean`	Устанавливает/возвращает наличие бланка маршрута документа. Используется вместе с объектом `RoutingSlip`
`HighlightChangesOnScreen`	`Boolean`	Устанавливает/возвращает видимое выделение изменений в совместно используемой книге
`HTMLProject`	`HTMLProject`	Только чтение. Возвращает объект, предоставляющий доступ к окну проекта в редакторе сценариев
`IsAddin`	`Boolean`	Устанавливает/возвращает информацию, является ли текущая книга надстройкой
`IsInplace`	`Boolean`	Только чтение. Возвращает информацию о том, выполняется ли редактирование книги в Microsoft Excel (`False`) или книга редактируется в виде внедренного объекта (`True`)
`KeepChangeHistory`	`Boolean`	Устанавливает/возвращает необходимость хранения списка изменений в совместно используемой книге
`ListChangesOnNewSheet`	`Boolean`	Устанавливает/возвращает необходимость вывода изменений в совместно используемой книге на отдельном листе
`Mailer`	`Mailer`	Только чтение. Используется только на платформе Macintosh
`MultiUserEditing`	`Boolean`	Только чтение. Возвращает информацию о том, используется ли книга совместно
`Name`	`String`	Только чтение. Возвращает имя файла книги
`Names`	`Names`	Только чтение. Возвращает коллекцию имен диапазонов в пределах книги
`Password`	`String`	Устанавливает/возвращает пароль, который должен вводиться для открытия книги
`PasswordEncryptionAlgorithm`	`String`	Только чтение. Возвращает алгоритм, который использовался в Excel для шифрования паролей книги
`PasswordEncryptionFileProperties`	`Boolean`	Только чтение. Возвращает информацию о шифровании свойств файла книги

Имя	*Тип возвращаемого значения*	*Описание*
PasswordEncryptionKeyLegth	Long	Только чтение. Возвращает длину ключа для алгоритма, который использовался в Excel для шифрования паролей книги
PasswordEncryptionProvider	String	Только чтение. Возвращает имя поставщика алгоритма шифрования, который использовался в Excel для шифрования паролей книги
Path	String	Только чтение. Возвращает путь к книге
PersonalViewListSettings	Boolean	Устанавливает/возвращает присутствие в пользовательском представлении книги фильтров и параметров сортировки для списков
PersonalViewPrintSettings	Boolean	Устанавливает/возвращает присутствие в пользовательском представлении книги параметров печати
PrecisionAsDisplayed	Boolean	Устанавливает/возвращает вывод точности чисел в ячейках. Используется при расчетах
ProtectStructure	Boolean	Только чтение. Возвращает необходимость сохранения порядка листов в книге
ProtectWindows	Boolean	Только чтение. Возвращает наличие защиты окон
PublishObjects	PublishObjects	Только чтение. Предоставляет доступ к объекту, который использовался для публикации объектов книги в виде Web-страниц
ReadOnly	Boolean	Только чтение. Возвращает состояние режима "Только чтение"
ReadOnlyRecommended	Boolean	Только чтение. Возвращает необходимость запроса у пользователя режима "Только чтение" при открытии книги
RemovePersonalInformation	Boolean	Устанавливает/возвращает возможность удаления персональной информации из книги
RevisionNumber	Long	Только чтение. Возвращает количество сохранений совместно используемой книги в процессе работы
Routed	Boolean	Только чтение. Результат перенаправления книги следующему получателю
RoutingSlip	RoutingSlip	Только чтение. Предоставляет доступ к объекту `RoutingSlip`, который используется для добавления бланка маршрута к книге. Применяется вместе со свойством `HasRoutingSlip`
Saved	Boolean	Устанавливает/возвращает наличие изменений, требующих сохранения
SaveLinkValues	Boolean	Устанавливает/возвращает необходимость сохранения значений, связанных с внешними источниками
Sheets	Sheets	Только чтение. Возвращает коллекцию листов в книге (`Chart` или `Worksheet`)
ShowConflictHistory	Boolean	Устанавливает/возвращает необходимость вывода листа, содержащего конфликты из совместно используемой книги

Имя	*Тип возвращаемого значения*	*Описание*
ShowPivot-TableField-List	Boolean	Устанавливает/возвращает необходимость отображения списка полей сводной таблицы
SmartTag-Options	SmartTagOp-tions	Только чтение. Возвращает операции, которые можно выполнять над смарт-тегами
Styles	Styles	Только чтение. Возвращает коллекцию стилей, доступных в книге
Template-RemoveExtData	Boolean	Устанавливает/возвращает необходимость удаления всех внешних ссылок при сохранении книги в виде шаблона
UpdateLinks	XlUpdateLinks	Устанавливает/возвращает параметры обновления внедренных ссылок OLE
UpdateRemote-References	Boolean	Устанавливает/возвращает необходимость обновления внешних ссылок
UserStatus	Variant	Только чтение. Возвращает имя текущего пользователя
VBASigned	Boolean	Только чтение. Возвращает наличие цифровой подписи проекта VBA для книги
VBProject	VBProject	Только чтение. Предоставляет доступ к проекту VBE и связанным с ним проектам
WebOptions	WebOptions	Только чтение. Возвращает объект, предоставляющий доступ к свойствам Web-документа книги
Windows	Windows	Только чтение. Возвращает коллекцию окон, составляющих книгу
Worksheets	Sheets	Только чтение. Возвращает коллекцию листов, составляющих книгу
WritePassword	String	Устанавливает/возвращает пароль защиты от записи в книгу
WriteReserved	Boolean	Только чтение. Возвращает возможность модификации книги
Write-ReservedBy	String	Только чтение. Возвращает имя пользователя, имеющего возможность модификации книги

Методы объекта Workbook

Имя	*Тип возвращаемого значения*	*Параметры*	*Описание*
Accept-Allchanges		[When], [Who], [Where]	Принимает все изменения, внесенные другими пользователями в совместно используемую книгу
Activate			Активизирует книгу
AddTo-Favorites			Добавляет книгу в папку Избранное (Favorites)
BreakLink		Name As String, Type As XlLinkType	Выполняет преобразование формул, связанных с другими источниками Excel или источниками OLE, в значения

Имя	*Тип возвращаемого значения*	*Параметры*	*Описание*
CanCheckIn	Boolean		Устанавливает/возвращает возможность передачи книги на сервер
ChangeFileAccess		Mode As XlFileAccess, [Write Password], [Notify]	Меняет разрешения на доступ к книге. Разрешения определяются значением параметра Mode. При необходимости в качестве значения параметра WritePassword можно указать пароль на запись. Для уведомления пользователя о невозможности доступа к файлу параметр Notify устанавливается в значение True
ChangeLink		Name As String, NewName As String, Type As XlLinkType	Меняет связь, указывающую на книгу в параметре Name, на связь, указывающую на книгу в параметре NewName. Параметр Type определяет тип связи (например, OLE, Excel).
CheckIn		[SaveChanges], [Comments], [MakePublic]	Включает или отменяет извлечение рабочей копии книги на сервере
Close		[SaveChanges], [Filename], [RouteWorkbook]	Закрывает книгу. Для автоматического сохранения изменений параметр SaveChanges устанавливается в значение True. Если параметр SaveChanges установлен в значение False, все изменения будут потеряны. В параметр Filename вносится имя файла, в котором сохраняется книга. Параметр RouteWorkbook используется для автоматической передачи книги следующему получателю.
DeleteNumberFormat		NumberFormat As String	Удаляет формат чисел (параметр NumberFormat) из книги
EndReview			Завершает рецензирование файла
ExclusiveAccess	Boolean		Предоставляет текущему пользователю исключительные права на доступ к совместно используемой книге

Имя	*Тип возвращаемого значения*	*Параметры*	*Описание*
FollowHyperlink		Address As String, [SubAddress], [NewWindow], [AddHistory], [ExtraInfo], [Method], [HeaderInfo]	Открывает соответствующее приложение с помощью адреса URL, который передается в качестве значения параметра Address. Для открытия гиперссылки в новом окне параметр NewWindow устанавливается в значение True. Для отправки дополнительной информации по гиперссылке (например, для страниц ASP) нужно воспользоваться параметрами ExtraInfo и Method. Для установки параметра Method используются константы MsoExtraInfoMethod
HighlightChangesOptions		[When], [Who], [Where]	Устанавливает/возвращает необходимость отображения изменений в совместно используемой книге (When), а именно: какие изменения должны отображаться (Who) и диапазон, в котором должны находиться изменения (Where). Для установки значения параметра When используются константы XlHighlighChangesTime
LinkInfo	Variant	Name As String, LinkInfo As XlLinkInfo, [Type], [EditionRef]	Возвращает дополнительную информацию о связи. Интересующая информация выбирается с помощью параметра LinkInfo. Ссылка выбирается с помощью параметра Name. Для установки параметра Type используются константы XlLinkInfoType (этот параметр позволяет выбрать тип возвращаемой связи)
LinkSources	Variant	[Type]	Возвращает массив связанных документов, редакций, серверов DDE и OLE в пределах книги. Для установки свойства Type используются константы XlLinkInfoType (этот параметр позволяет выбрать тип возвращаемой связи)
MergeWorkbook		Filename	Объединяет изменения из книги, на которую указывает параметр Filename, с текущей книгой

Имя	*Тип возвращаемого значения*	*Параметры*	*Описание*
NewWindow	Windows		Открывает новое окно с текущей книгой
OpenLinks		Name As String, [ReadOnly], [Type]	Открывает связь Name и поддерживающие документы. Для открытия документа в режиме “только чтения” параметр ReadOnly устанавливается в значение True. Для установки параметра Type используются константы XlLinkInfoType (этот параметр позволяет выбрать тип возвращаемой связи)
PivotCaches	PivotCaches		Возвращает коллекцию кэшей сводной таблицы
Post		[DestName]	Публикует книгу в общей папке на сервере Microsoft Exchange
PrintOut		[From], [To], [Copies], [Preview], [ActivePrinter], [PrintToFile], [Collate], [PrToFileName]	Печатает книгу. Принтер, количество копий, порядок и необходимость режима предварительного просмотра определяются с помощью параметров. Кроме этого, листы можно печатать в файл. Для этого используются параметры PrintToFile и PrToFileName. Параметры From и To могут использоваться для определения печатаемого диапазона
PrintPreview		[EnableChanges]	Выводит текущую книгу в режиме предварительного просмотра. Для отключения кнопок Поля и Параметры страницы параметр EnableChanges устанавливается в значение False. При этом пользователь не сможет модифицировать параметры страницы в режиме предварительного просмотра
Protect		[Password], [Structure], [Windows]	Защищает книгу от модификации пользователем. При этом параметру Password можно присвоить пароль для защиты книги. Для защиты относительного расположения листов параметр Structure устанавливается в значение Ture. Для защиты окон книги параметр Windows устанавливается в значение True

Имя	*Тип возвращаемого значения*	*Параметры*	*Описание*
ProtectSharing		[Filename], [Password], [WriteResPassword], [ReadOnlyRecommended], [CreateBackup], [SharingPassword]	Защищает и сохраняет книгу для совместного использования. Имя сохраняемого файла передается в качестве значения параметра Filename. Необязательный пароль можно указать в качестве значения параметров Password, WriteResPassword и SharingPassword. Для вывода сообщения при каждом открытии книги параметр ReadOnlyRecommended устанавливается в значение True. Для создания резервной копии сохраняемого файла параметр CreateBackup устанавливается в значение True
PurgeChangeHistoryNow		Days As Long, [SharingPassword]	Удаляет записи из журнала изменений совместно используемой книги. Параметр Days определяет давность удаляемых записей (в днях). При этом может потребоваться предоставление значения параметра SharingPassword
RecheckSmartTags			Выполняет проверку смарт-тегов. Все нераспознанные ранее данные распознаются в результате этой проверки
RefreshAll			Обновляет все данные из внешних источников данных
RejectAllChanges		[When], [Who], [Where]	Отменяет все изменения в совместно используемой книге
ReloadAs		Encoding As MsoEncoding	Повторно открывает книгу из Web-страницы. Кодировка страницы указывается в качестве значения параметра Encoding
RemoveUser		Index As Long	Отключает пользователя (пользователь выбирается с помощью параметра Index) от совместно используемой книги
ReplyAll			Отвечает всем получателям совместно используемой книги. Используется только на платформе Macintosh
ReplyWithChanges		[ShowMessage]	Отправляет по электронной почте уведомление о завершении рецензирования автору книги

Имя	*Тип возвращаемого значения*	*Параметры*	*Описание*
ResetColors			Сбрасывает цветовую палитру книги
Route			Передает книгу с использованием бланка маршрута
RunAutoMacros		Which As XlRunAutoMacro	Запускает автоматический макрос, на который указывает параметр Which
Save			Сохраняет книгу
SaveAs		Filename, FileFormat, Password, WriteResPassword, ReadOnlyRecommended, CreateBackup, AccessMode As XlSaveAsAccessMode, [ConflictResolution], [AddToMru], [TextCodepage], [TextVisualLayout], [Local]	Сохраняет книгу в файле, имя которого указывается в качестве значения параметра Filename. Тип сохраняемого файла определяется параметром FileFormat. Файл может быть сохранен с паролем. Для этого указываются значения параметров Password и WriteResPassword. Для вывода рекомендации о доступе только для чтения при каждом открытии файла параметр ReadOnlyRecommended устанавливается в значение True. Для создания резервной копии сохраняемого файла параметр CreateBackup устанавливается в значение True. Параметр AccessMode позволяет указать тип доступа к книге (например, xlShared, xlExclusive). Для выбора метода разрешения конфликтов можно воспользоваться параметром ConflictResolution. Для добавления книги в список недавно открывавшихся файлов параметр AddToMru устанавливается в значение True
SaveCopyAs		[Filename]	Сохраняет копию книги в файле, имя которого указывается в качестве значения параметра Filename
SendForReview		[Recipients], [Subject], [ShowMessage], [IncludeAttachment]	Отправляет книгу на адрес электронной почты указанных получателей для рецензирования

Имя	*Тип возвращаемого значения*	*Параметры*	*Описание*
`SendMail`		`Recipients, [Subject], [ReturnReceipt]`	Отправляет книгу с помощью принятой по умолчанию системы электронной почты. Получатель или получатели, а также тема сообщения с книгой, могут указываться в виде значений параметров. Для получения подтверждения о доставке параметр `ReturnReceipt` устанавливается в значение `True`
`SendMailer`		`FileFormat, Priority As XlPriority`	Используется только на платформе Macintosh
`SetLinkOnData`		`Name As String, [Procedure]`	Запускает процедуру из параметра `Procedure` при каждом обновлении связи DDE или OLE из параметра `Name`
`SetPasswordEncryptionOptions`		`[PasswordEncryptionProvider], [PasswordEncryptionAlgorithm], [PasswordEncryptionKeyLength], [PasswordEncryptionFileProperties]`	Устанавливает параметры шифрования книги с использованием пароля
`Unprotect`		`[Password]`	Снимает с книги защиту с помощью пароля
`UnprotectSharing`		`[SharingPassword]`	Снимает с книги защиту от совместного доступа и сохраняет книгу
`UpdateFromFile`			Повторно загружает текущую книгу из файла, если файл имеет более позднюю дату изменения, чем открытая книга
`UpdateLink`		`[Name], [Type]`	Обновляет связь, указанную с помощью параметра `Name`. Параметр `Type` и константы `XlLinkInfoType` позволяют выбрать тип возвращаемой связи
`WebPagePreview`			Включает режим предварительного просмотра книги перед преобразованием в Web-страницу

События объекта Workbook

Имя	*Параметры*	*Описание*
Activate		Возникает при активизации книги
AddinInstall		Возникает при открытии книги в виде надстройки
AddinUninstall		Возникает при удалении книги, открытой в качестве надстройки
BeforeClose	Cancel As Boolean	Возникает перед закрытием книги. Для отмены закрытия параметр Cancel устанавливается в значение True
BeforePrint	Cancel As Boolean	Возникает перед печатью книги. Для отмены печати параметр Cancel устанавливается в значение True
BeforeSave	SaveAsUI As Boolean, Cancel As Boolean	Возникает перед сохранением книги. Для отмены сохранения параметр Cancel устанавливается в значение True. Для вывода диалогового окна Сохранить как (Save As) параметр SaveAsUI устанавливается в значение True
Deactivate		Возникает при потере фокуса
NewSheet	Sh As Object	Возникает при создании нового листа. Новый лист передается в процедуру обработки события в качестве значения параметра Sh
Open		Возникает при открытии книги
PivotTable-Close-Connection	ByVal Target As PivotTable	Возникает при закрытии подключения отчета сводной таблицы к источнику данных. Параметр Target содержит выделенную сводную таблицу
PivotTable-OpenConnection	ByVal Target As PivotTable	Возникает при открытии подключения отчета сводной таблицы к источнику данных. Параметр Target содержит выделенную сводную таблицу
SheetActivate	Sh As Object	Возникает при активизации листа в пределах книги. Лист передается в процедуру обработки события в качестве значения параметра Sh
SheetBefore-DoubleClick	Sh As Object, Target As Range, Cancel As Boolean	Возникает перед двойным щелчком на листе. Лист и потенциальная точка двойного щелчка передаются в процедуру обработки события в качестве значений параметров. Для отмены двойного щелчка параметр Cancel устанавливается в значение True
SheetBefore-RightClick	Sh As Object, Target As Range, Cancel As Boolean	Возникает перед щелчком правой кнопкой мыши на листе. Лист и потенциальная точка щелчком правой кнопкой мыши передаются в процедуру обработки события в качестве значений параметров. Для отмены щелчка правой кнопкой мыши параметр Cancel устанавливается в значение True
SheetCalculate	Sh As Object	Возникает при пересчете листа. Пересчитываемый лист передается в процедуру обработки события

Имя	*Параметры*	*Описание*
`SheetChange`	`Sh As Object, Target As Range`	Возникает при изменении содержимого ячейки на любом листе в пределах книги, например при вводе новых данных, очистке содержимого ячейки и удалении строк или столбцов. Не возникает при добавлении строк или столбцов
`SheetDeactivate`	`Sh As Object`	Возникает при потере листом фокуса. Лист передается в процедуру обработки события в качестве значения параметра
`SheetFollowHyperlink`	`Sh As Object, Target As Hyperlink`	Возникает при щелчке пользователя на гиперссылке в пределах листа. В процедуру обработки события передается лист и гиперссылка, на которой щелкнул пользователь
`SheetPivotTableUpdate`	`ByVal Sh As Object, Target As PivotTable`	Возникает при обновлении листа с отчетом сводной таблицы
`SheetSelectionChange`	`Sh As Object, Target As Range`	Возникает при выделении другой ячейки на листе. В процедуру обработки события передается новый диапазон и лист, на котором произошло событие
`WindowActivate`	`Wn As Window`	Возникает при активизации окна книги (окно размещается на переднем плане поверх остальных книг). Книга и окно передаются в процедуру обработки события
`WindowDeactivate`	`Wn As Window`	Возникает при потере фокуса окном. Книга и окно передаются в процедуру обработки события
`WindowResize`	`Wn As Window`	Возникает при изменении размера окна книги. Книга и окно передаются в процедуру обработки события

Пример: объект Workbook и коллекция Workbooks

Дополнительная информация об использовании этих объектов приводилась в главе 4.

Объект Worksheet и коллекция Worksheets

В коллекции `Worksheets` хранятся листы из книги. Объект `Workbook` всегда является родительским для коллекции `Worksheets`, в которой хранятся только листы электронных таблиц. Для доступа к объектам `Worksheet` в коллекции `Worksheets` можно воспользоваться свойством `Item`. В качестве параметра свойства `Item` используется имя листа или номер, описывающий положение листа в коллекции (нумерация выполняется слева направо).

Объект `Worksheet` предоставляет доступ ко всем атрибутам листа в Excel. Это касается как форматирования листа, так и других его свойств. Кроме того, объект `Worksheet` предоставляет события, которые могут обрабатываться программно.

Кроме типичных атрибутов, коллекция `Worksheet` предоставляет дополнительные свойства и методы, показанные в следующей таблице.

Свойства и методы коллекции Worksheets

Имя	*Тип возвращаемого значения*	*Описание*
Count	Long	Только чтение. Возвращает количество листов в коллекции
HPageBreaks	HPAgeBreaks	Только чтение. Возвращает коллекцию горизонтальных разрывов страницы, связанных с коллекцией Worksheets
VPageBreaks	VPageBreaks	Только чтение. Возвращает коллекцию вертикальных разрывов страницы, связанных с коллекцией Worksheets
Visible	Variant	Устанавливает/возвращает видимость листов из коллекции. Если установить это свойство в значение xlVeryHidden, пользователь не сможет сделать видимыми листы из коллекции
Add		Метод. Параметры: [Before], [After], [Count], [Type]. Добавляет лист в коллекцию. Положение листа в коллекции определяется с помощью параметров. Можно указать лист перед новым листом (параметр Before) или лист после нового листа (параметр After). Параметр Count позволяет указать количество создаваемых листов
Copy		Метод. Параметры: [Before], [After]. Добавляет в коллекцию копию активного листа. Положение добавляемой копии определяется параметрами
Delete		Метод. Удаляет все листы из коллекции
FillAcross-Sheets		Метод. Параметры: Range As Range, Type As XlFill-With. Копирует диапазон из параметра Range во все листы из коллекции. Параметр Type позволяет определить копируемую часть диапазона (например, xlFillWithContents, xlFillWithFormulas)
Move		Метод. Параметры: [Before], [After]. Перемещает текущий лист в позицию, указанную с помощью параметров
PrintPreview		Метод. Параметры: [EnableChanges]. Выводит текущий лист в режиме предварительного просмотра. Для отключения кнопок Поля и Параметры страницы параметр EnableChanges устанавливается в значение True. При этом пользователь не сможет изменить параметры страницы
PrintOut		Метод. Параметры: [From], [To], [Copies], [Preview], [ActivePrinter], [PrintToFile], [Collate], [PrToFileName]. Печатает листы из коллекции. Принтер, количество копий, порядок и необходимость предварительного просмотра определяются с помощью параметров. Кроме этого, листы можно печатать в файл. Для этого нужно воспользоваться параметрами PrintToFile и PrToFileName. Параметры From и To позволяют определить диапазон печатаемых страниц
Select		Метод. Параметры: [Replace]. Выделяет текущий лист в коллекции

Общие свойства объекта Worksheet

Определения свойств Application, Creator и Parent приводились в начале приложения.

Свойства объекта Worksheet

Имя	*Тип возвращаемого значения*	*Описание*
AutoFilter	AutoFilter	Только чтение. Возвращает объект AutoFilter, если включена фильтрация
AutoFilterMode	Boolean	Устанавливает/возвращает вывод списка автофильтра на листе
Cells	Range	Только чтение. Возвращает ячейки текущего листа
Circular-Reference	Range	Только чтение. Возвращает диапазон ячеек, содержащий первую циклическую ссылку в пределах листа
CodeName	String	Только чтение. Возвращает имя листа, установленное на этапе проектирования в редакторе VBE
Columns	Range	Только чтение. Возвращает диапазон столбцов в текущем листе
Comments	Comments	Только чтение. Возвращает коллекцию комментариев на листе
Consolidation-Function	XlConsolidation-Function	Только чтение. Возвращает тип консолидации, которая используется на листе (например, xlSum, xlMax, xlAverage)
Consolidation-Options	Variant	Только чтение. Возвращает одномерный массив, содержащий три бинарных элемента. Первый элемент описывает необходимость использования меток в верхней строке. Второй — необходимость использования меток в левом столбце. Третий — необходимость создания ссылок на исходные данные
Consolidation-Sources	Variant	Только чтение. Возвращает массив строк, описывающий исходные листы для текущей консолидации
Custom-Properties	Custom-Properties	Только чтение. Возвращает информацию идентификатора, связанную с листом
DisplayPage-Breaks	Boolean	Устанавливает/возвращает необходимость отображения разрывов страницы
DisplayRight-ToLeft	Boolean	Устанавливает/возвращает необходимость отображения содержимого листа справа налево. Используется только в языках, применяющих запись справа налево
EnableAuto-Filter	Boolean	Устанавливает/возвращает необходимость вывода списков автофильтра при защите пользовательского интерфейса листа
Enable-Calculation	Boolean	Устанавливает/возвращает возможность автоматического пересчета листа при необходимости
Enable-Outlining	Boolean	Устанавливает/возвращает возможность использования символов структуры при защите пользовательского интерфейса листа

Имя	*Тип возвращаемого значения*	*Описание*
`EnablePivotTable`	`Boolean`	Устанавливает/возвращает возможность отображения элеметнов управления сводной таблицы и связанных с ними операций при защите пользовательского интерфейса листа
`EnableSelection`	`XlEnableSelection`	Устанавливает/возвращает возможность выделения объектов при защите листа (например, `xlNoSelection`, `xlNoRestrictions`)
`FilterMode`	`Boolean`	Только чтение. Возвращает состояние режима фильтрации листа
`HPageBreaks`	`HPageBreaks`	Только чтение. Возвращает коллекцию с горизонтальными разрывами страницы на листе
`Hyperlinks`	`Hyperlinks`	Только чтение. Возвращает коллекцию гиперссылок на листе
`Index`	`Long`	Только чтение. Возвращает положение текущего листа в родительской коллекции
`ListObjects`	`Range`	Возвращает объект `Range`
`MailEnvelope`	`MsoEnvelope`	Устанавливает/возвращает заголовок сообщения электронной почты для документа
`Name`	`String`	Устанавливает/возвращает имя листа
`Names`	`Names`	Только чтение. Возвращает коллекцию именованных диапазонов на листе
`Next`		Только чтение. Возвращает следующий лист в книге (слева направо) в виде объекта
`Outline`	`Outline`	Только чтение. Возвращает объект, позволяющий управлять структурой листа
`PageSetup`	`PageSetup`	Только чтение. Возвращает объект, позволяющий управлять параметрами страницы для листа
`Previous`		Только чтение. Возвращает предыдущий лист в книге (справа налево) в виде объекта
`ProtectContents`	`Boolean`	Только чтение. Возвращает состояние защиты листа и всего его содержимого
`ProtectDrawingObjects`	`Boolean`	Только чтение. Возвращает возможность модификации фигур на листе (для модификации свойство `ProtectDrawingObjects` устанавливается в значение `False`)
`Protection`	`Protection`	Только чтение. Возвращает параметры защиты листа
`ProtectionMode`	`Boolean`	Только чтение. Возвращает состояние защиты пользовательского интерфейса. Даже если на пользовательский интерфейс листа установлена защита, остается возможность доступа к связанному с листом коду VBA
`ProtectScenarios`	`Boolean`	Только чтение. Возвращает состояние защиты сценариев на листе
`QueryTables`	`QueryTables`	Только чтение. Возвращает коллекцию таблиц запроса, связанных с листом

Имя	*Тип возвращаемого значения*	*Описание*
Range	Range	Только чтение. Параметры: Cell1, [Cell2]. Возвращает объект Range, на который указывают параметры Cell1 и Cell2
Rows	Range	Только чтение. Возвращает объект Range, содержащий строки текущего листа
Scripts	Scripts	Только чтение. Возвращает коллекцию связанных с листом сценариев VBScript (обычно эти сценарии используются в Web-документах)
ScrollArea	String	Устанавливает строку ссылки в формате A1, описывающую прокручиваемый диапазон листа. Выделение ячеек за пределами этого диапазона невозможно
Shapes	Shapes	Только чтение. Возвращает все геометрические фигуры в пределах листа
SmartTags	SmartTags	Только чтение. Возвращает идентификатор указанной ячейки
StandardHeight	Double	Только чтение. Возвращает принятую по умолчанию высоту строк на листе (в пунктах)
StandardWidth	Double	Только чтение. Возвращает принятую по умолчанию ширину столбцов на листе (в пунктах)
Tab	Tab	Только чтение. Возвращает объект Tab для выделенной диаграммы или листа
TransitionExp-Eval	Boolean	Устанавливает/возвращает необходимость оценки выражений на листе с помощью правил Lotus 1-2-3
Transition-FormEntry	Boolean	Устанавливает/возвращает возможность ввода формул с помощью правил Lotus 1-2-3
Type	XlSheetType	Только чтение. Возвращает тип листа (например, xlWorksheet, xlExcel4MacroSheet, xlExcel4IntlMacroSheet)
UsedRange	Range	Только чтение. Возвращает используемый диапазон листа
Visible	XlSheet-Visibility	Устанавливает/возвращает видимость текущего листа. Установка свойства в значение xlVeryHidden приведет к невозможности включения видимости через пользовательский интерфейс
VPageBreaks	VPageBreaks	Только чтение. Возвращает коллекцию всех вертикальных разрывов страницы на листе

Методы листа Worksheet

Имя	*Тип возвращаемого значения*	*Параметры*	*Описание*
Activate			Активизирует лист
Calculate			Вычисляет все формулы на листе
ChartObjects		[Index]	Возвращает объект диаграммы (ChartObject) или коллекцию объектов диаграммы (ChartObjects)
CheckSpelling		[CustomDictionary], [IgnoreUppercase], [AlwaysSuggest], [SpellLang]	Проверяет орфографию текста на листе. Параметр CustomDictionary позволяет выбрать собственный словарь. Параметр IgnoreUppercase позволяет игнорировать слова, состоящие только из прописных букв. Параметр AlwaysSuggest заставляет Excel предлагать слова для замены
CircleInvalid			Выделяет некорректные записи на листе
ClearArrows			Удаляет все стрелки зависимостей на листе
ClearCircles			Удаляет все выделения вокруг некорректных записей на листе
Copy		[Before], [After]	Добавляет новую копию листа. Положение новой копии определяется параметрами Before и After
Delete			Удаляет лист
Evaluate	Variant	Name	Оценивает выражение из параметра Name так же, как и при вводе выражения в ячейку листа
Move		[Before], [After]	Перемещает лист в положение, описанное параметрами
OLEObjects		[Index]	Возвращает один объект OLE (OLEObject) или коллекцию объектов OLE (OLEObjects), связанных с листом
Paste		[Destination], [Link]	Вставляет в лист содержимое буфера обмена. Диапазон назначения можно определить с помощью параметра Destination. Для создания связи с источником вставляемых данных параметр Link устанавливается в значение True. Может использоваться один из параметров
PasteSpecial		[Format], [Link], [DisplayAsIcon], [IconFileName], [IconIndex], [IconLabel], [NoHTMLFormatting]	Вставляет содержимое буфера обмена в текущий лист. Формат данных в буфере обмена определяется с помощью параметра Format. Для создания связи с исходными данными параметр Link устанавливается в значение True. Для отображения вставляемых данных в виде пиктограммы параметр DisplayAsIcon устанавливается в значение True. При этом параметры IconFileName, IconIndex и IconLabel позволяют определить пиктограмму и метку пиктограммы. Диапазон назначения уже должен присутствовать на листе

Имя	*Тип возвращаемого значения*	*Параметры*	*Описание*
PivotTables		[Index]	Возвращает один отчет сводной таблицы (PivotTable) или коллекцию отчетов сводной таблицы (PivotTables), связанных с листом
PivotTableWizard	PivotTable	[SourceType], [SourceData], [TableDestination], [TableName], [RowGrand], [ColumnGrand], [SaveData], [HasAutoFormat], [AutoPage], [Reserved], [BackgroundQuery], [OptimizeCache], [PageFieldOrder], [PageFieldWrapCount], [ReadData], [Connection]	Создает отчет сводной таблицы. Параметр SourceType определяет тип используемых исходных данных, которые передаются в качестве значения параметра SourceData. Для установки значения параметра SourceType используются константы XlPivotTableSourceType. В качестве значения параметра TableDestination указывается диапазон родительского листа, в котором будет расположен отчет сводной таблицы. Параметр TableName содержит имя нового отчета. Для вывода итогов для строк и столбцов параметры RowGrand и ColumnGrand устанавливаются в значение True. Для автоматического форматирования отчета при обновлении или модификации параметр HasAutoFormat устанавливается в значение True. Для автоматического создания полей страниц для консолидации необходимо воспользоваться параметром AutoPage. Для асинхронного опроса источников данных параметр BackgroundQuery устанавливается в значение True. Для оптимизации кэша при создании параметр OptimizeCache устанавливается в значение True. Параметр PageFieldOrder и константы XlOrder позволяют определить способ добавления новых полей страниц. Параметр PageFieldWrapCount определяет количество полей страниц в каждом столбце и строке. Для копирования данных из внешнего источника в кэш сводной таблицы параметр ReadData устанавливается в значение True. И наконец, необходимо воспользоваться параметром Connection для определения строки подключения ODBC для кэша сводной таблицы
PrintOut		[From], [To], [Copies], [Preview], [ActivePrinter], [PrintToFile], [Collate], [PrToFileName]	Печатает лист. Принтер, количество копий, порядок и необходимость предварительного просмотра задаются с помощью параметров. Кроме этого, лист можно напечатать в файл. Для этого используются параметры PrintToFile и PrToFileName. Параметры From и To определяют диапазон печатаемых страниц

Имя	*Тип возвращаемого значения*	*Параметры*	*Описание*
PrintPreview		[EnableChanges]	Выводит лист в режиме предварительного просмотра. Для отключения кнопок Поля и Параметры страницы параметр EnableChanges устанавливается в значение False. При этом изменение параметров страницы через пользовательский интерфейс невозможно
Protect		[Password], [DrawingObjects], [Contents], [Scenarios], [UserInterfaceOnly], [AllowFormattingCells], [AllowFormattingColumns], [AllowFormattingRows], [AllowInsertingColumns], [AllowInsertingRows], [AllowInsertingHyperlinks], [AllowDeletingColumns], [AllowDeletingRows], [AllowSorting], [AllowFiltering], [AllowUsingPivotTables]	Защищает лист от внесения изменений. При этом можно указать чувствительный к регистру пароль. Кроме этого, можно выбрать защиту только геометрических фигур (DrawingObjects), всего содержимого (Contents) или защиту только пользовательского интерфейса (UserInterfaceOnly)
ResetAllPageBreaks			Сбрасывает все разрывы страниц на листе
SaveAs		Filename As String, [FileFormat], [Password], [WriteResPassword], [ReadOnlyRecommended], [CreateBackup], [AddToMru], [TextCodepage], [TextVisualLayout], [Local]	Сохраняет лист в файле, имя которого передается в качестве значения параметра Filename. Тип сохраняемого файла определяется параметром FileFormat. При необходимости сохраняемый файл защищается паролем. Для этого можно воспользоваться параметрами WriteResPassword и Password. Для вывода рекомендации о доступе только для чтения при каждом открытии файла параметр ReadOnlyRecommended устанавливается в значение True. Для создания резервной копии сохраняемого файла параметр CreateBackup устанавливается в значение True. Для добавления листа в список недавно открытых файлов параметр AddToMru устанавливается в значение True

Имя	*Тип возвращаемого значения*	*Параметры*	*Описание*
Scenarios		[Index]	Возвращает один сценарий (Scenario) или коллекцию сценариев (Scenarios) для листа
Select		[Replace]	Выделяет лист
SetBack-ground-Picture		Filename As String	Устанавливает фоновый рисунок листа. Имя файла с рисунком указывается в качестве значения параметра Filename
ShowAll-Data			Отображает все отфильтрованные данные
ShowData-Form			Выводит форму данных, которая является частью листа
Unprotect		[Password]	Снимает защиту с листа. Если лист был защищен паролем, пароль передается в качестве значения параметра
XmlData-Query	Range	[XPath], [Selection-Namespaces], [Map]	Представляет ячейки, отображающиеся на определенный запрос XPath
XmlMap-Query	Range	[XPath], [Selection-Namespaces], [Map]	Представляет ячейки, отображающиеся на определенный запрос XPath

События объекта Worksheet

Имя	*Тип параметры*	*Описание*
Activate		Возникает при активизации листа
Before-DoubleClick	Target As Range, Cancel As Boolean	Возникает перед двойным щелчком на листе. В качестве значения параметра Target в процедуру обработки события передается ячейка, ближайшая к точке двойного щелчка. Для отмены события двойного щелчка необходимо установить параметр Cancel в значение True
Before-RightClick	Cancel As Boolean	Возникает перед щелчком правой кнопкой мыши на листе. В качестве значения параметра Target в процедуру обработки события передается ячейка, ближайшая к точке щелчка правой кнопкой мыши. Для отмены щелчка правой кнопкой мыши необходимо установить параметр Cancel в значение True
Calculate		Возникает после пересчета листа
Change	Target As Range	Возникает при изменении значений в ячейках листа. Измененный диапазон передается в процедуру обработки события в виде значения параметра Target
Deactivate		Возникает при деактивизации листа
Follow-Hyperlink	Target As Hyperlink	Возникает при щелчке на гиперссылке в пределах листа. Гиперссылка передается в процедуру обработки события в качестве значения параметра Target
PivotTable-Update	ByVal Target As PivotTable	Возникает при обновлении отчета сводной таблицы

Имя	Тип параметры	Описание
SelectionChange	Target As Range	Возникает при изменении выделения на листе. Диапазон нового выделения передается в процедуру обработки события в виде значения параметра Target

Пример: объект Worksheet и коллекция Worksheets

Дополнительная информация об использовании объекта Worksheet приводится в главе 4.

Объект WorksheetFunction

В объекте WorksheetFunction хранятся все функции листа Excel. Этот объект предоставляет доступ из кода VBA к функциям листа Excel. Родительским для объекта WorksheetFunction является объект Application.

Общие свойства объекта WorksheetFunction

Определения свойств Application, Creator и Parent приводились в начале приложения.

Методы объекта WorksheetFunction

Имя	Тип возвращаемого значения	Параметры	Описание
Acos	Double	Arg1 As Double	Возвращает арккосинус значения аргумента Arg1. Значение аргумента Arg1 должно находиться в пределах от –1 до 1
Acosh	Double	Arg1 As Double	Возвращает обратный гиперболический косинус значения аргумента Arg1. Значение аргумента Arg1 должно быть больше или равно 1
And	Boolean	Arg1, [Arg2], ... [Arg30]	Возвращает значение True, если все аргументы (от Arg1 до Arg30) равны True
Asc	String	Arg1 As String	Возвращает символ половинной длины, который соответствует символу полной длины в строке Arg1. Эта функция отличается от функции VBA Asc, которая возвращает код ASCII для первого символа в строке
Asin	Double	Arg1 As Double	Возвращает арксинус значения аргумента Arg1. Значение аргумента Arg1 должно находиться в пределах от –1 до 1
Asinh	Double	Arg1 As Double	Возвращает обратный гиперболический синус значения аргумента Arg1

Имя	*Тип возвращаемого значения*	*Параметры*	*Описание*
Atan2	Double	Arg1 As Double, Arg2 As Double	Возвращает арктангенс координат X и Y, передаваемых с помощью аргументов Arg1 и Arg2
Atanh	Double	Arg1 As Double	Возвращает обратный гиперболический арктангенс значения аргумента Arg1. Значение аргумента Arg1 должно находиться в пределах от –1 до 1
AveDev	Double	Arg1, [Arg2], ... [Arg30]	Возвращает среднее абсолютных отклонений для числовых параметров от Arg1 до Arg30
Average	Double	Arg1, [Arg2], ... [Arg30]	Возвращает среднее значение чисел из параметров от Arg1 до Arg30
BahtText	String	Arg1 As Double	Возвращает число на тайском языке с суффиксом "Baht.". Число указывается в качестве значения параметра Arg1
BetaDist	Double	Arg1 As Double, Arg2 As Double, Arg3 As Double, [Arg4], [Arg5]	Возвращает интегральную функцию бета-распределения. В аргументе Arg1 содержится оцениваемое число. В аргументе Arg2 — значение альфа-параметра распределения. В аргументе Arg3 — значение бета-параметра распределения. В аргументах Arg4 и Arg5 могут передаваться верняя и нижняя границы интервала для аргумента Arg1
BetaInv	Double	Arg1 As Double, Arg2 As Double, Arg3 As Double, [Arg4], [Arg5]	Возвращает обратную интегральную функцию бета-распределения. Аргумент Arg1 содержит вероятность распределения. Аргумент Arg2 — значение альфа-параметра распределения. Аргумент Arg3 — значение бета-параметра распределения. В аргументах Arg4 и Arg5 могут передаваться верняя и нижняя границы интервала для оцениваемого числа
BinomDist	Double	Arg1 As Double, Arg2 As Double, Arg3 As Double, Arg4 As Boolean	Возвращает отдельное значение биномиального распределения вероятности. В аргументе Arg1 передается количество успешных испытаний. В аргументе Arg2 содержится общее количество испытаний. В аргументе Arg3 содержится вероятность успешности отдельного испытания. Аргумент Arg4 определяет необходимость возврата кумулятивной функции распределения (True) или массовой функции вероятности (False)

Имя	*Тип возвращаемого значения*	*Параметры*	*Описание*
Ceiling	Double	Arg1 As Double, Arg2 As Double	Возвращает ближайшее к значению аргумента Arg1 число, кратное значению аргумента Arg2 с округлением вверх
ChiDist	Double	Arg1 As Double, Arg2 As Double	Возвращает одностороннюю вероятность распределения Хи-квадрат. В аргументе Arg1 содержится оцениваемое число. В аргументе Arg2 — количество степеней свободы
ChiInv	Double	Arg1 As Double, Arg2 As Double	Возвращает значение, обратное односторонней вероятности распределения Хи-квадрат. В аргументе Arg1 содержится вероятность. В аргументе Arg2 — количество степеней свободы
ChiTest	Double	Arg1, Arg2	Возвращает тест на независимость для распределения Хи-квадрат. В аргументе Arg1 передается диапазон с данными, который будет сравниваться с ожидаемыми значениями. В аргументе Arg2 передается диапазон с ожидаемыми значениями
Choose	Variant	Arg1, Arg2, [Arg3], ... [Arg30]	Возвращает одно из значений аргументов (с Arg2 по Arg30). Номер аргумента указывается в качестве значения аргумента Arg1. Например, если аргумент Arg1 равен 2, то возвращается значение аргумента Arg3
Clean	String	Arg1 As String	Возвращает строку из аргумента Arg1 без непечатаемых символов
Combin	Double	Arg1 As Double, Arg2 As Double	Возвращает общее количество комбинаций группы элементов. Размер группы указывается в качестве значения аргумента Arg2. Общее количество элементов указывается в качестве значения аргумента Arg1
Confidence	Double	Arg1 As Double, Arg2 As Double, Arg3 As Double	Возвращает диапазон по одну сторону среднего значения выборки для заданного среднего значения всего множества. Аргумент Arg1 содержит значение параметра альфа для определения уровня уверенности. Аргумент Arg2 содержит стандартное отклонение для диапазона данных. Аргумент Arg3 содержит размер выборки
Correl	Double	Arg1, Arg2	Возвращает коэфициент корреляции для массивов из аргументов Arg1 и Arg2. В качестве параметров метода можно указывать диапазоны ячеек

Имя	*Тип возвращаемого значения*	*Параметры*	*Описание*
Cosh	Double	Arg1 As Double	Возвращает гиперболический косинус значения Arg1
Count	Double	Arg1, [Arg2], ... [Arg30]	Возвращает количество числовых значений среди аргументов. Аргументам можно присваивать как значения, так и ссылки на диапазоны
CountA	Double	Arg1, [Arg2], ... [Arg30]	Возвращает количество непустых значений аргументов. Аргументам можно присваивать как значения, так и ссылки на диапазоны
CountBlank	Double	Arg1 As Range	Возвращает количество пустых ячеек в диапазоне из аргумента Arg1
CountIf	Double	Arg1 As Range, Arg2	Подсчитывает количество ячеек в диапазоне Arg1, удовлетворяющих критерий из аргумента Arg2
Covar	Double	Arg1, Arg2	Возвращает ковариацию двух массивов или диапазонов из аргументов Arg1 и Arg2
CritBinom	Double	Arg1 As Double, Arg2 As Double, Arg3 As Double	Возвращает наименьшее значение, для которого интегральное биномиальное распределение больше или равно заданному критерию. Аргумент Arg1 содержит количество испытаний Бернулли, Arg2 — вероятность успеха в каждом испытании. Аргумент Arg3 содержит значение критерия
DAverage	Double	Arg1 As Range, Arg2, Arg3	Возвращает среднее значение столбца, указанного с помощью аргумента Arg2 в диапазоне ячеек из диапазона Arg1. Аргумент Arg3 содержит критерий для выбора строк при подсчете среднего значения
Days360	Double	Arg1, Arg2, [Arg3]	Возвращает количество дней между датами в аргументах Arg1 и Arg2 (Arg1 - Arg2). Если аргумент Arg3 установлен в значение True, используется европейский метод подсчета. Если аргумент Arg3 установлен в значение False или отсутствует, используется американский метод подсчета
Db	Double	Arg1 As Double, Arg2 As Double, Arg3 As Double, Arg4 As Double, [Arg5]	Возвращает амортизацию имущества за заданный период, используя метод постоянного учета амортизации. Аргумент Arg1 содержит исходную стоимость амортизации. Аргумент Arg2 содержит финальную стоимость списания. Аргумент Arg3 содержит количество периодов амортизации. Аргумент Arg4 указывает конкретный период из аргумента Arg3. Аргумент Arg5 содержит количество месяцев в первый год амортизации

Имя	*Тип возвращаемого значения*	*Параметры*	*Описание*
Dbcs	String	Arg1 As String	Возвращает строку двухбайтовых символов для заданной строки ASCII. Является противоположностью функции Asc
DCount	Double	Arg1 As Range, Arg2, Arg3	Возвращает количество ячеек, соответствующих критерию из аргумента Arg3. Аргумент Arg1 содержит диапазон строк и столбцов, а аргумент Arg2 — имя поля или подсчитываемое количество
DCountA		Arg1 As Range, Arg2, Arg3	Возвращает количество непустых ячеек, соответствующих критерию из аргумента Arg3. Аргумент Arg1 содержит диапазон строк и столбцов, а аргумент Arg2 — имя поля или подсчитываемое количество
Ddb	Double	Arg1 As Double, Arg2 As Double, Arg3 As Double, Arg4 As Double, [Arg5]	Возвращает величину амортизации имущества для указанного периода при использовании метода двукратного учета амортизации. Аргумент Arg1 содержит исходную стоимость амортизации. Аргумент Arg2 содержит стоимость списания (стоимость на момент завершения амортизации). Аргумент Arg3 содержит количество периодов амортизации. Аргумент Arg4 указывает на определенный период из аргумента Arg3. Аргумент Arg5 может содержать скорость амортизации
Degrees	Double	Arg1 As Double	Выполняет преобразование радианов из аргумента Arg1 в градусы
DevSq	Double	Arg1, [Arg2], ... [Arg30]	Возвращает сумму квадратов отклонений для аргументов от Arg1 до Arg30
DGet	Variant	Arg1 As Range, Arg2, Arg3	Возвращает значение ячейки, соответствующее критерию из аргумента Arg3. Аргумент Arg1 содержит диапазон подсчитываемых строк и столбцов, а аргумент Arg2 позволяет выбрать имя поля или подсчитываемое количество. Критерий должен описывать единственную ячейку
DMax	Double	Arg1 As Range, Arg2, Arg3	Возвращает максимальное значение, соответствующее критерию Arg3. Аргумент Arg1 содержит диапазон подсчитываемых строк и столбцов. Аргумент Arg2 используется для выбора имени поля или подсчитываемого количества

Имя	*Тип возвращаемого значения*	*Параметры*	*Описание*
DMin	Double	Arg1 As Range, Arg2, Arg3	Возвращает минимальное значение, соответствующее критерию Arg3. Аргумент Arg1 содержит диапазон подсчитываемых строк и столбцов. Аргумент Arg2 используется для выбора имени поля или подсчитываемого количества
Dollar	String	Arg1 As Double, [Arg2]	Преобразовывает число в текст, используя денежный формат. Аргумент Arg1 содержит число. Аргумент Arg2 содержит количество десятичных знаков
DProduct	Double	Arg1 As Range, Arg2, Arg3	Возвращает произведение значений, соответствующих критерию из аргумента Arg3. Аргумент Arg1 содержит диапазон подсчитываемых строк и столбцов. Аргумент Arg2 используется для выбора имени поля или подсчитываемого количества
DStDev	Double	Arg1 As Range, Arg2, Arg3	Возвращает прогноз стандартного отклонения для значений, соответствующих критерию из аргумента Arg3. Аргумент Arg1 содержит диапазон подсчитываемых строк и столбцов. Аргумент Arg2 используется для выбора имени поля или подсчитываемого количества
DStDevP	Double	Arg1 As Range, Arg2, Arg3	Возвращает прогноз стандартного отклонения для значений, соответствующих критерию из аргумента Arg3 в предположении о том, что предоставлено все множество значений. Аргумент Arg1 содержит диапазон подсчитываемых строк и столбцов. Аргумент Arg2 используется для выбора имени поля или подсчитываемого количества
DSum	Double	Arg1 As Range, Arg2, Arg3	Возвращает сумму значений, соответствующих критерию из аргумента Arg3. Аргумент Arg1 указывает диапазон подсчитываемых строк и столбцов. Аргумент Arg2 содержит имя поля или подсчитываемое количество
DVar	Double	Arg1 As Range, Arg2, Arg3	Возвращает вариацию значений, соответствующих критерию из аргумента Arg3. Аргумент Arg1 указывает диапазон подсчитываемых строк и столбцов. Аргумент Arg2 используется для выбора имени поля или подсчитываемого количества

Имя	*Тип возвращаемого значения*	*Параметры*	*Описание*
DVarP	Double	Arg1 As Range, Arg2, Arg3	Возвращает вариацию значений, соответствующих критерию из аргумента Arg3 в предположении о том, что предоставлено все множество значений. Аргумент Arg1 указывает на диапазон подсчитываемых строк и столбцов. Аргумент Arg2 используется для выбора имени поля или подсчитываемого количества
Even	Double	Arg1 As Double	Выполняет преобразование числа из аргумента Arg1 в ближайшее четное целое число с округлением вверх
ExponDist	Double	Arg1 As Double, Arg2 As Double, Arg3 As Boolean	Возвращает экспоненциальное распределение значения. Аргумент Arg1 содержит значение функции. Аргумент Arg2 содержит значение параметра лямбда. Для возврата интегрального распределения параметр Arg3 устанавливается в значение True. Для получения плотности вероятности аргумент Arg3 устанавливается в значение False
Fact	Double	Arg1 As Double	Возвращает факториал аргумента Arg1
FDist	Double	Arg1 As Double, Arg2 As Double, Arg3 As Double	Возвращает F-распределение вероятности. Аргумент Arg1 содержит значение для оценки функции. Аргумент Arg2 содержит числитель степеней свободы. Аргумент Arg3 содержит знаменатель степеней свободы
Find	Double	Arg1 As String, Arg2 As String, [Arg3]	Находит текст из аргумента Arg1 в тексте из аргумента Arg2. Возвращает положение первого символа найденного текста. Аргумент Arg3 указывает позицию начала поиска в значении аргумента Arg2
FindB	Double	Arg1 As String, Arg2 As String, [Arg3]	Находит текст из аргумента Arg1 в тексте из аргумента Arg2. Возвращает положение первого символа найденного текста. Аргумент Arg3 указывает позицию начала поиска (в байтах) в значении аргумента Arg2
FInv	Double	Arg1 As Double, Arg2 As Double, Arg3 As Double	Возвращает обратное значение для F-распределения вероятности. Аргумент Arg1 содержит вероятность, связанную с интегральным F-распределением. Аргумент Arg2 содержит числитель степеней свободы. Аргумент Arg3 содержит знаменатель степеней свободы

Имя	*Тип возвращаемого значения*	*Параметры*	*Описание*
`Fisher`	`Double`	`Arg1 As Double`	Возвращает преобразование Фишера для значения из аргумента `Arg1`
`FisherInv`	`Double`	`Arg1 As Double`	Возвращает результат обратного преобразования Фишера для значения аргумента `Arg1`
`Fixed`	`String`	`Arg1 As Double, [Arg2], [Arg3]`	Округляет число из аргумента `Arg1` до количества десятичных знаков, указанных в аргументе `Arg2`, и возвращает число в виде строки. Для добавления запятых в возвращаемый текст аргумент `Arg3` устанавливается в значение `True`. Если аргумент `Arg3` установить в значение `False`, запятые в возвращаемый текст не добавляются
`Floor`	`Double`	`Arg1 As Double, Arg2 As Double`	Возвращает ближайшее к значению аргумента `Arg1` кратное значению `Arg2` число, округленное в сторону 0
`Forecast`	`Double`	`Arg1 As Double, Arg2, Arg3`	Вычисляет будущее значение `y` по текущему значению `x` (аргумент `Arg1`) на основе существующих пар значений. Аргумент `Arg2` содержит массив или диапазон с известными значениями `y`. Аргумент `Arg3` содержит массив или диапазон с известными значениями `x`
`Frequency`	`Variant`	`Arg1, Arg2`	Возвращает распределение частот в виде вертикального массива. Массив значений находится в аргументе `Arg1`, а массив интервалов — в аргументе `Arg2`. Кроме этого, аргументы `Arg1` и `Arg2` могут содержать ссылки на диапазоны
`FTest`	`Double`	`Arg1, Arg2`	Возвращает результат F-теста для массивов из аргументов `Arg1` и `Arg2`. Кроме этого, аргументы `Arg1` и `Arg2` могут содержать ссылки на диапазоны
`Fv`	`Double`	`Arg1 As Double, Arg2 As Double, Arg3 As Double, [Arg4], [Arg5]`	Возвращает будущее значение вклада для заданного периода времени. Аргумент `Arg1` содержит процентную ставку для периода. Аргумент `Arg2` — общее количество периодов выплат. Аргумент `Arg3` — размер выплат за определенный период. Аргумент `Arg4` — начальное значение. Аргумент `Arg5` определяет необходимость выплат на конец (0) или на начало (1) периода

Имя	*Тип возвращаемого значения*	*Параметры*	*Описание*
GammaDist	Double	Arg1 As Double, Arg2 As Double, Arg3 As Double, Arg4 As Boolean	Возвращает гамма-распределение. Аргумент Arg1 содержит оцениваемое значение. Аргумент Arg2 — значение параметра альфа для распределения. Аргумент Arg3 — значение параметра бета для распределения. Для получения интегрального распределения аргумент Arg4 устанавливается в значение True. Для получения плотности вероятности аргумент Arg4 устанавливается в значение False
GammaInv	Double	Arg1 As Double, Arg2 As Double, Arg3 As Double	Возвращает обратное гамма-распределение. Аргумент Arg1 содержит вероятность, связанную с гамма-распределением. Аргумент Arg2 — параметр альфа для распределения. Аргумент Arg3 — параметр бета для распределения
GammaLn	Double	Arg1 As Double	Возвращает значение натурального логарифма гамма-функции для значения аргумента Arg1
GeoMean	Double	Arg1, [Arg2], ... [Arg30]	Возвращает среднее геометрическое для значений аргументов с Arg1 по Arg30. Аргументы могут содержать ссылки на диапазоны
Growth	Variant	Arg1, [Arg2], [Arg3], [Arg4]	Возвращает прогнозируемые значения y (аргумент Arg1) в соответствии с экспоненциальным трендом новых значений x (аргумент Arg3). Аргумент Arg2 может использоваться для предоставления последовательности существующих значений x. Если аргумент Arg4 установить в значение False, часть 'b' уравнения будет равна 1
HarMean	Double	Arg1, [Arg2], ... [Arg30]	Возвращает среднее гармоническое для значений аргументов с Arg1 по Arg30. Аргументы могут содержать ссылки на диапазоны
HLookup	Variant	Arg1, [Arg2], [Arg3], [Arg4]	Выполняет поиск значения аргумента Arg1 в массиве таблицы (или по ссылке на диапазон). Аргумент Arg2 указывает на первую строку. Аргумент Arg3 содержит номер строки массива, включающей искомое значение. Для поиска похожих данных аргумент Arg4 устанавливается в значение True. Для поиска точного совпадения аргумент Arg4 устанавливается в значение False

Имя	*Тип возвращаемого значения*	*Параметры*	*Описание*
HypGeomDist	Double	Arg1 As Double, Arg2 As Double, Arg3 As Double, Arg4 As Double	Возвращает гипергеометрическое распределение вероятности. Аргумент Arg1 содержит количество успешных испытаний в выборке. Аргумент Arg2 — размер выборки. Аргумент Arg3 — количество успешных испытаний во всем множестве, а аргумент Arg4 — размер всего множества
Index	Variant	Arg1, Arg2 As Double, [Arg3], [Arg4]	Возвращает ячейку или массив ячеек из аргумента Arg1. Номер строки содержится в аргументе Arg2. Номер столбца содержится в аргументе Arg3. Кроме того, может возвращать ячейку или диапазон ячеек из аргумента Arg1. Номер строки содержится в аргументе Arg2. Номер столбца содержится в аргументе Arg3. Если аргумент Arg1 состоит из нескольких областей, аргумент Arg4 используется для выбора области
Intercept	Double	Arg1, Arg2	Возвращает точку пересечения координат по оси x и оси y. Аргумент Arg1 содержит массив известных значений y, а аргумент Arg2 — известных значений x
Ipmt	Double	Arg1 As Double, Arg2 As Double, Arg3 As Double, Arg4 As Double, [Arg5], [Arg6]	Возвращает величину выплат прибыли на вложения за данный период. Аргумент Arg1 содержит процентную ставку за период. Аргумент Arg2 — период, за который производятся выплаты. Аргумент Arg3 — общее количество периодов выплат. Аргумент Arg4 — начальное значение. Аргумент Arg5 — значение выплат, которое необходимо получить в будущем. Аргумент Arg6 определяет необходимость выплат в конце (0) или в начале (1) периода
Irr	Double	Arg1, [Arg2]	Возвращает внутреннюю ставку доходности для массива значений из аргумента Arg1. Аргумент Arg2 содержит предположительное значение функции Irr
IsErr	Boolean	Arg1	Сообщает, содержит ли ячейка из аргумента Arg1 ошибочное значение (#N/A)
IsError	Boolean	Arg1	Сообщает, содержит ли ячейка из аргумента Arg1 любое ошибочное значение

Имя	*Тип возвращаемого значения*	*Параметры*	*Описание*
IsLogical	Boolean	Arg1	Сообщает, содержит ли ячейка из аргумента Arg1 логическое значение
IsNA	Boolean	Arg1	Сообщает, содержит ли ячейка из аргумента Arg1 значение #N/A
IsNonText	Boolean	Arg1	Сообщает, содержит ли ячейка из аргумента Arg1 нетекстовое значение
IsNumber	Boolean	Arg1	Сообщает, содержит ли ячейка из аргумента Arg1 числовое значение
Ispmt	Double	Arg1 As Double, Arg2 As Double, Arg3 As Double, Arg4 As Double	Возвращает проценты, выплачиваемые за определенный инвестиционный период. Используется для обеспечения совместимости. Аргумент Arg1 содержит процентную ставку для периода. Аргумент Arg2 — период, для которого необходимо подсчитать процент. Аргумент Arg3 — общее количество периодов. Аргумент Arg4 — начальное значение
IsText	Boolean	Arg1	Сообщает, содержит ли ячейка из аргумента Arg1 текстовое значение
Kurt	Double	Arg1, Arg2], ... [Arg30]	Возвращает эксцесс значений из аргументов с Arg1 по Arg30. Кроме этого, аргумент Arg1 может содержать ссылку на диапазон ячеек
Large	Double	Arg1, Arg2 As Double	Возвращает наибольшее значение из массива или диапазона ячеек, на который указывает аргумент Arg1. Аргумент Arg2 содержит номер значения (например, второе наибольшее значение, третье наибольшее значение)
LinEst	Variant	Arg1, [Arg], [Arg3], [Arg4]	Возвращает массив, описывающий прямую, которая лучше всего соответствует известным значениям y (аргумент Arg1) и x (Arg2). Для присвоения значения 0 части уравнения 'b' аргумент Arg3 устанавливается в значение False. Аргумент Arg4 устанавливается в значение True для получения дополнительной статистики
Ln	Double	Arg1 As Double	Возвращает натуральный логарифм значения аргумента Arg1
Log	Double	Arg1 As Double, [Arg2]	Возвращает логарифм числа Arg1 по основанию, указанному в качестве значения аргумента Arg2. По умолчанию аргумент Arg2 равен 10
Log10	Double	Arg1 As Double	Возвращает логарифм значения Arg1 по основанию 10

Имя	*Тип возвращаемого значения*	*Параметры*	*Описание*
LogEst	Variant	Arg1, [Arg2], [Arg3], [Arg4]	Возвращает массив, описывающий кривую, которая наилучшим образом описывает известные значения y (аргумент Arg1) и x (аргумент Arg2). Для присвоения значения 0 части уравнения 'b' аргумент Arg3 устанавливается в значение False. Аргумент Arg4 устанавливается в значение True для получения дополнительной статистики
LogInv	Double	Arg1 As Double, Arg2 As Double, Arg3 As Double	Возвращает обратное логарифмическое интегральное распределение. Аргумент Arg1 содержит инвертируемую вероятность. Аргумент Arg2 — среднее значение для ln(*значение*). Аргумент Arg4 — стандартное отклонение для ln(*значение*)
LogNormDist	Double	Arg1 As Double, Arg2 As Double, Arg3 As Double	Возвращает интегральное логарифмическое нормальное распределение для значения аргумента Arg1. Аргумент Arg2 содержит среднее значение для ln(Arg1). Аргумент Arg3 содержит стандартное отклонение для ln(Arg1)
Lookup	Variant	Arg1, Arg2, [Arg3]	Выполняет поиск значения аргумента Arg1 в строке или столбце из аргумента Arg2. Возвращает значение из той же позиции массива из аргумента Arg3
Match	Double	Arg1, Arg2, [Arg3]	Возвращает относительное положение элемента в массиве из аргумента Arg2. Элемент должен совпадать со значением аргумента Arg1. Для выбора типа совпадения используется аргумент Arg3
Max	Double	Arg1, [Arg2], ... [Arg30]	Возвращает максимальное значение аргументов от Arg1 до Arg30. Кроме того, аргумент Arg1 может содержать диапазон ячеек
MDeterm	Double	Arg1	Возвращает определитель матрицы, которая описывается массивом из аргумента Arg1. Кроме этого, аргумент Arg1 может содержать диапазон ячеек. Ячейки не могут содержать текст или быть пустыми. Количество строк и стобцов должны совпадать
Median	Double	Arg1, [Arg2], ... [Arg30]	Возвращает значение медианы для чисел из аргументов с Arg1 по Arg30. Кроме этого, аргумент Arg1 может содержать диапазон ячеек

Имя	*Тип возвращаемого значения*	*Параметры*	*Описание*
Min	Double	Arg1, [Arg2], ... [Arg30]	Возвращает наименьшее значение среди аргументов с Arg1 по Arg30. Кроме того, аргумент Arg1 может содержать диапазон ячеек
MInverse	Variant	Arg1	Возвращает обратную матрицу для массива из аргумента Arg1. Кроме этого, аргумент Arg1 может содержать диапазон ячеек. Ячейки матрицы не могут содержать текст и быть пустыми. Количество строк и столбцов должно совпадать
MIrr	Double	Arg1, Arg2 As Double, Arg3 As Double	Возвращает модифицированную ставку доходности для последовательности значений из аргумента Arg1. Аргумент Arg2 содержит размер выплат. Аргумент Arg3 содержит размер выплат при реинвестировании денежного потока
MMult	Variant	Arg1, Arg2	Возвращает произведение матриц из двух массивов Arg1 и Arg2. Количество столбцов в аргументе Arg1 должно совпадать с количеством строк в аргументе Arg2. Аргументы Arg1 и Arg2 могут содержать диапазон ячеек. Ячейки не могут содержать текст и быть пустыми
Mode	Double	Arg1, [Arg2], ... [Arg30]	Возвращает наиболее часто встречающееся число из значений аргументов с Arg1 по Arg30. Кроме этого, аргумент Arg1 может содержать диапазон ячеек
NegBinom-Dist	Double	Arg1 As Double, Arg2 As Double, Arg3 As Double	Возвращает отрицательное биномиальное распределение аргументов. Аргумент Arg1 содержит количество неудачных испытаний. Аргумент Arg2 — граничное количество успешных испытаний. Аргумент Arg3 — вероятность успеха при единичном испытании
NormDist	Double	Arg1 As Double, Arg2 As Double, Arg3 As Double, Arg4 As Boolean	Возвращает интегральное нормальное распределение для значения (аргумент Arg1), среднего (аргумент Arg2) и стандартного отклонения (аргумент Arg3). Для получения интегрального распределения аргумент Arg4 устанавливается в значение True. Для получения плотности распределения аргумент Arg4 устанавливается в значение False

Имя	*Тип возвращаемого значения*	*Параметры*	*Описание*
NormSDist	Double	Arg1 As Double	Возвращает стандартное нормальное интегральное распределение для указанной вероятности (аргумент Arg1)
NormSInv	Double	Arg1 As Double	Возвращает обратное значение стандартного нормального распределения для указанной вероятности (аргумент Arg1)
NPer	Double	Arg1 As Double, Arg2 As Double, Arg3 As Double, [Arg4], [Arg5]	Возвращает общее количество периодов выплат для данного вклада. Аргумент Arg1 содержит процент для периода. Аргумент Arg2 — объем выплат за каждый период. Аргумент Arg3 — начальное значение. Аргумент Arg4 — интересующий размер вклада в будущем. Аргумент Arg5 определяет необходимость выплат в конце (0) или в начале (1) периода
Npv	Double	Arg1 As Double, Arg2, [Arg3], ... [Arg30]	Возвращает чистую приведенную стоимость инвестиции, используя ставку дисконтирования (аргумент Arg1), а также стоимость будущих выплат и поступлений (аргументы с Arg2 по Arg30)
Odd	Double	Arg1 As Double	Выполняет округление в сторону увеличения значения аргумента Arg1 до ближайшего целого нечетного числа
Or	Boolean	Arg1, [Arg2], ... [Arg30]	Возвращает значение True, если любой из аргументов имеет значение True
Pearson	Double	Arg1, Arg2	Возвращает коэффициент корреляции Пирсона для массива значений. Аргумент Arg1 содержит массив независимых значений, а аргумент Arg2 — массив зависимых значений. Аргументы Arg1 и Arg2 могут содержать ссылки на диапазоны
Percentile	Double	Arg1, Arg2 As Double	Возвращает k-тую персентиль для значений из диапазона ячеек или массива, хранящегося в аргументе Arg1. Значение k предоставляется через аргумент Arg2
PercentRank	Double	Arg1, Arg2 As Double, [Arg3]	Возвращает процентную норму значения в множестве данных. Аргумент Arg1 содержит диапазон ячеек или массив с данными. Аргумент Arg2 содержит значение. Аргумент Arg3 определяет количество значащих цифр возвращаемой процентной нормы

Имя	*Тип возвращаемого значения*	*Параметры*	*Описание*
Permut	Double	Arg1 As Double, Arg2 As Double	Возвращает общее количество перестановок для заданного числа объектов. Аргумент Arg1 содержит общее число объектов. Аргумент Arg2 содержит число выбранных объектов Arg2
Phonetic	String	Arg1 As Range	Возвращает фонетические символы для текстовой строки из аргументов Arg1
Pi	Double		Возвращает значение числа π (3,14) с точностью до 15 десятичных знаков
Pmt	Double	Arg1 As Double, Arg2 As Double, Arg3 As Double, [Arg4], [Arg5]	Возвращает величину выплат по заему. Аргумент Arg1 содержит процентную ставку за период. Аргумент Arg2 — количество выплат по заему. Аргумент Arg3 — начальное значение. Аргумент Arg4 — значение, которое необходимо достигнуть в будущем. Аргумент Arg5 определяет необходимость выплат в конце (0) или в начале (1) периода
Poisson	Double	Arg1 As Double, Arg2 As Double, Arg3 As Boolean	Возвращает распределение Пусассона. Аргумент Arg1 содержит количество событий. Аргумент Arg2 содержит ожидаемое числовое значение. Для получения интегральной вероятности аргумент Arg3 устанавливается в значение True. Для получения плотности вероятности аргумент Arg3 устанавливается в значение False
Power	Double	Arg1 As Double, Arg2 As Double	Возвращает степень для основания Arg1 и показателя Arg2
Ppmt	Double	Arg1 As Double, Arg2 As Double, Arg3 As Double, Arg4 As Double, [Arg5], [Arg6]	Возвращает размер выплат по вкладу для заданного периода. Аргумент Arg1 содержит процент за период. Аргумент Arg2 указывает на интересующий период. Аргумент Arg3 содержит общее количество выплат. Аргумент Arg4 — начальное значение. Аргумент Arg5 — интересующую стоимость в будущем. Аргумент Arg6 определяет необходимость выплат в конце (0) или в начале (1) периода
Prob	Double	Arg1, Arg2, Arg3 As Double, [Arg4]	Возвращает вероятность попадания массива Arg1 и связанного с ним массива Arg2 между нижним (аргумент Arg3) и верхним (аргумент Arg4) пределами

Имя	*Тип возвращаемого значения*	*Параметры*	*Описание*
Product	Double	Arg1, [Arg2], ... [Arg30]	Возвращает произведение значений всех аргументов с Arg1 по Arg30
Proper	String	Arg1 As String	Делает первую букву каждого слова из аргумента Arg1 заглавной и все остальные буквы прописными
Pv	Double	Arg1 As Double, Arg2 As Double, Arg3 As Double, [Arg4], [Arg5]	Возвращает приведенную стоимость инвестиции. Аргумент Arg1 содержит значение процента для периода. Аргумент Arg2 — количество выплат по вкладу. Аргумент Arg3 — размер выплат за период. Аргумент Arg4 — интересующее в будущем значение. Аргумент Arg5 определяет необходимость выплат на конец (0) или на начало (1) периода
Quartile	Double	Arg1, Arg2 As Double	Возвращает квартиль для массива из аргумента Arg1. Аргумент Arg2 может содержать 0 (минимальное значение), 1 (первая квартиль), 2 (вторая квартиль), 3 (третья квартиль) или 4 (максимальное значение)
Radians	Double	Arg1 As Double	Выполняет преобразование значения аргумента Arg1 из градусов в радианы и возвращает новое значение
Rank	Double	Arg1 As Double, Arg2 As Range, [Arg3]	Возвращает ранг значения аргумента Arg1 в диапазоне из аргумента Arg2. Аргумент Arg3 описывает ранжировку диапазона из аргумента Arg1
Rate	Double	Arg1 As Double, Arg2 As Double, Arg3 As Double, [Arg4], [Arg5], [Arg6]	Возвращает размер процента за период для достижения определенной стоимости. Аргумент Arg1 содержит общее количество выплат. Аргумент Arg2 — размер выплат за период. Аргумент Arg3 — начальное значение. Аргумент Arg4 — стоимость в будущем, которой необходимо достичь. Аргумент Arg5 определяет необходимость выплат в конце (0) или в начале (1) периода
Replace	String	Arg1 As String, Arg2 As Double, Arg3 As Double, Arg4 As String	Заменяет фрагмент текста из аргумента Arg1 текстом из аргумента Arg4. Аргумент Arg2 указывает на начало заменяемого фрагмента. Аргумент Arg3 определяет количество заменяемых символов

Имя	*Тип возвращаемого значения*	*Параметры*	*Описание*
ReplaceB	String	Arg1 As String, Arg2 As Double, Arg3 As Double, Arg4 As String	Заменяет фрагмент текста из аргумента Arg1 текстом из аргумента Arg4. Аргумент Arg2 указывает на начало заменяемого фрагмента. Аргумент Arg3 определяет количество заменяемых байтов
Rept	String	Arg1 As String, Arg2 As Double	Повторяет строку из аргумента Arg1 несколько раз (количество раз указывается в качестве значения аргумента Arg2)
Roman	String	Arg1 As Double, [Arg2]	Возвращает запись значения аргумента Arg1 римскими цифрами. Аргумент Arg2 определяет стиль римских цифр. Значение 0 или True описывает классический стиль. Значение 4 или False описывает упрощенный стиль. Можно считать, что значения от 0 до 4 описывают разные степени упрощения
Round	Double	Arg1 As Double, Arg2 As Double	Возвращает значение Arg1, округленное до количества цифр, указанного с помощью аргумента Arg2
RoundDown	Double	Arg1 As Double, Arg2 As Double	Возвращает значение Arg1, округленное до количества цифр, указанного с помощью аргумента Arg2. Значение округляется в сторону уменьшения
RoundUp	Double	Arg1 As Double, Arg2 As Double	Возвращает значение Arg1, округленное до количества цифр, указанного с помощью аргумента Arg2. Значение округляется в сторону увеличения
RSq	Double	Arg1, Arg2	Возвращает квадрат коэффициента корреляции Пирсона. Аргумент Arg1 содержит массив значений y. Аргумент Arg2 содержит массив значений x. Кроме этого, аргументы Arg1 и Arg2 также могут содержать ссылки на диапазоны

Имя	*Тип возвращаемого значения*	*Параметры*	*Описание*
RTD	Variant	progID As Variant, server As Variant, topic1 As Variant, [topic2], [topic3], [topic4], [topic5], [topic6], [topic7], [topic8], [topic9], [topic10], [topic11], [topic12], [topic13], [topic14], [topic15], [topic16], [topic17], [topic18], [topic19], [topic20], [topic21], [topic22], [topic23], [topic24], [topic25], [topic26], [topic27], [topic28]	Подключается к источнику данных для получения данных реального времени. Аргумент progID содержит программный идентификатор сервера реального времени. Аргумент server — имя сервера. Аргумент topic1 — интересующую тему
Search	Double	Arg1 As String, Arg2 As String, [Arg3]	Находит текст из аргумента Arg1 в строке из аргумента Arg2. Метод возвращает положение найденного текста в строке. Аргумент Arg3 указывает позицию начала поиска в аргументе Arg2
SearchB	Double	Arg1 As String, Arg2 As String, [Arg3]	Находит текст из аргумента Arg1 в строке из аргумента Arg2. Метод возвращает положение найденного текста в строке. Аргумент Arg3 указывает позицию начала поиска (в байтах) в аргументе Arg2
Sinh	Double	Arg1 As Double	Возвращает гиперболический синус аргумента Arg1
Skew	Double	Arg1, [Arg2], ... [Arg30]	Возвращает ассиметрию распределения для чисел из аргументов с Arg1 по Arg30. Аргументы могут содержать ссылки на диапазоны

Имя	*Тип возвращаемого значения*	*Параметры*	*Описание*
Sln	Double	Arg1 As Double, Arg2 As Double, Arg3 As Double	Возвращает величину амортизации, рассчитанную линейным методом. Аргумент Arg1 содержит стоимость актива. Аргумент Arg3 — остаточную стоимость. Аргумент Arg2 — количество периодов амортизации
Slope	Double	Arg1, Arg2	Возвращает наклон линии линейной регрессии, проходящей через точки данных, которые описываются значениями X из аргумента Arg1 и Y из аргумента Arg2
Small	Double	Arg1, Arg2 As Double	Возвращает k-тое наименьшее значение массива или диапазона из аргумента Arg1 (например, второе наименьшее, третье наименьшее). Аргумент Arg2 содержит значение k
Standardize	Double	Arg1 As Double, Arg2 As Double, Arg3 As Double	Возвращает нормализованное значение на основе распределения. Аргумент Arg1 содержит интересующее значение. Аргумент Arg2 — среднее значение. Аргумент Arg3 — стандартное отклонение
StDev	Double	Arg1, [Arg2], ... [Arg30]	Возвращает прогнозируемое стандартное отклонение для значений с Arg1 по Arg30. Кроме этого, аргументы могут содержать ссылки на диапазоны
StDevP	Double	Arg1, [Arg2], ... [Arg30]	Возвращает стандартное отклонение для всех значений с Arg1 по Arg30. Кроме этого, аргументы могут содержать ссылки на диапазоны
StEyx	Double	Arg1, Arg2	Возвращает стандартную ошибку прогнозируемых значений y для каждого значения x в регрессии. Аргумент Arg1 содержит известные значения y. Аргумент Arg2 содержит значения x
Substitute	String	Arg1 As String, Arg2 As String, Arg3 As String, [Arg4]	Заменяет все экземпляры текста из аргумента Arg2 текстом из аргумента Arg3. Аргумент Arg1 содержит модифицируемую строку. Аргумент Arg4 позволяет выбрать заменяемый экземпляр текста

Имя	*Тип возвращаемого значения*	*Параметры*	*Описание*
`Subtotal`	`Double`	`Arg1 As Double, Arg2 As Range, [Arg3], ... [Arg30]`	Возвращает итоги для диапазонов или ссылок на диапазоны, указанные в аргументах с `Arg2` по `Arg30`. Аргумент `Arg1` содержит число, описывающее тип используемой функции для подсчета итогов. Действительные номера функций находятся в пределах от 1 до 10. Эти числа представляют функции `Average`, `Count`, `CountA`, `Max`, `Min`, `Product`, `StcDev`, `StDevP`, `Sum`, `Var` и `VarP`
`Sum`	`Double`	`Arg1, [Arg2], ... [Arg30]`	Возвращает сумму значений из аргументов с `Arg1` по `Arg30`
`SumIf`	`Double`	`Arg1 As Range, Arg2, [Arg3]`	Возвращает сумму ячеек в диапазоне из аргумента `Arg1`. Ячейки должны соответствовать критерию из аргумента `Arg2`. Диапазон для хранения суммы можно указать с помощью аргумента `Arg3`. Строки или столбцы в аргументах `Arg1` и `Arg3` должны совпадать
`SumProduct`	`Double`	`Arg1, [Arg2], ... [Arg30]`	Умножает соответствующие элементы из массивов с `Arg1` по `Arg30` и возвращает сумму произведений. Размерность массивов должна совпадать
`SumSq`	`Double`	`Arg1, [Arg2], ... [Arg30]`	Возвращает сумму квадратных корней значений аргументов с `Arg1` по `Arg30`
`SumX2MY2`	`Double`	`Arg1, Arg2`	Вычисляет разность квадратов соответствующих элементов массивов `Arg1` и `Arg2` и возвращает сумму всех новых элементов
`SumX2PY2`	`Double`	`Arg1, Arg2`	Вычисляет сумму квадратов соответствующих элементов массивов `Arg1` и `Arg2` и возвращает сумму всех новых элементов
`SumXMY2`	`Double`	`Arg1, Arg2`	Вычисляет разность соответствующих элементов массивов `Arg1` и `Arg2`, возводит разность в квадрат и возвращает сумму новых элементов
`Syd`	`Double`	`Arg1 As Double, Arg2 As Double, Arg3 As Double, Arg4 As Double`	Возвращает величину амортизации актива за заданный период, рассчитанную по методу “сумма годовых чисел”. Аргумент `Arg1` содержит начальную стоимость. Аргумент `Arg2` — стоимость списания. Аргумент `Arg3` — количество периодов амортизации. Аргумент `Arg4` указывает на интересующий период

Имя	*Тип возвращаемого значения*	*Параметры*	*Описание*
Tanh	Double	Arg1 As Double	Возвращает гиперболический тангенс аргумента Arg1
Tdist	Double	Arg1 As Double, Arg2 As Double, Arg3 As Double	Возвращает вероятность t-распределения Стьюдента для значения из аргумента Arg1. Аргумент Arg2 содержит количество степеней свободы. Аргумент Arg3 устанавливается в значение 1 для получения одностороннего распределения. Аргумент Arg3 устанавливается в значение 2 для получения двустороннего распределения
Text	String	Arg1, Arg2 As String	Выполняет преобразование значения аргумента Arg1 в текстовую строку с использованием формата из аргумента Arg2
Tinv	Double	Arg1 As Double, Arg2 As Double	Возвращает обратное t-распределение Стьюдента для указанных вероятности (аргумента Arg1) и количества степеней свободы (аргумент Arg2)
Transpose	Variant	Arg1	Выполняет транспонирование диапазона из аргумента Arg1. Возвращает новый диапазон
Trend	Variant	Arg1, [Arg2], [Arg3], [Arg4]	Возвращает значения в соответствии с линейным трендом. Метод аппроксимирует прямой линией известные значения y (аргумент Arg1) и x (аргумент Arg2), а также новые значения x (аргумент Arg3). Аргумент Arg4 устанавливается в значение False для присвоения части уравнения 'b' значения 0
Trim	String	Arg1 As String	Возвращает строку из аргумента Arg1 без вступительных и завершающих пробелов
TrimMean	Double	Arg1, Arg2 As Double	Возвращает среднее значение внутренней области массива данных из аргумента Arg1. Аргумент Arg2 позволяет указать исключаемую часть массива

Имя	*Тип возвращаемого значения*	*Параметры*	*Описание*
Ttest	Double	Arg1, Arg2, Arg3 As Double, Arg4 As Double	Возвращает вероятность, соответствующую критерию Стьюдента, на основании двух наборов данных из аргумента Arg1 и Arg2. Аргумент Arg3 устанавливается в значение 1 для получения одностороннего распределения. Аргумент Arg3 устанавливается в значение 2 для получения двустороннего распределения. Для выбора типа t-теста: парный, гомоскедастический и гетероскедастический, аргумент Arg4 устанавливается в значения 1, 2 или 3
USDollar	String	Arg1 As Double, Arg2 As Double	Возвращает строку в валютном формате, соответствующую числу из аргумента Arg1. Количество десятичных знаков указывается в качестве значения аргумента Arg2
Var	Double	Arg1, [Arg2], ... [Arg30]	Возвращает прогнозируемую вариацию на основании значений аргументов с Arg1 по Arg30. Кроме этого, аргументы могут содержать ссылки на диапазоны
VarP	Double	Arg1, [Arg2], ... [Arg30]	Возвращает вариацию на основе значений аргументов с Arg1 по Arg30 с использованием всего множества. Кроме этого, аргументы могут содержать ссылки на диапазоны
Vdb	Double	Arg1 As Double, Arg2 As Double, Arg3 As Double, Arg4 As Double, Arg5 As Double, [Arg6], [Arg7]	Возвращает амортизацию с использованием метода ее двукратного учета (если не выбран другой метод) для указанных периодов. Аргумент Arg1 содержит начальную стоимость. Аргумент Arg2 — остаточную стоимость. Аргумент Arg3 — количество периодов амортизации. Аргумент Arg4 указывает на начальный период амортизации, а аргумент Arg5 — на конечный. Аргумент Arg6 содержит скорость амортизации. Для вычисления амортизации с помощью метода двукратного учета аргумент Arg7 устанавливается в значение True. Для переключения на метод постоянного учета амортизации аргумент Arg7 устанавливается в значение False

Имя	*Тип возвращаемого значения*	*Параметры*	*Описание*
`Vlookup`	`Variant`	`Arg1, Arg2, Arg3, [Arg4]`	Выполняет поиск значения аргумента `Arg1` в массиве таблицы (или в диапазоне) из аргумента `Arg2`. Аргумент `Arg3` содержит номер столбца, включающего совпадающее значение. Для приблизительного поиска аргумент `Arg4` устанавливается в значение `True`. Для поиска точных значений аргумент `Arg4` устанавливается в значение `False`
`Weekday`	`Double`	`Arg1, [Arg2]`	Возвращает номер дня недели для даты из аргумента `Arg1`. Аргумент `Arg2` содержит день начала недели
`Weibull`	`Double`	`Arg1 As Double, Arg2 As Double, Arg3 As Double, Arg4 As Boolean`	Возвращает распределение Вейбула для значения аргумента `Arg1`. Аргумент `Arg2` содержит значение параметра распределения альфа. Аргумент `Arg3` содержит значение параметра распределения бета. Для получения интегральной вероятности аргумент `Arg4` устанавливается в значение `True`. Для получения плотности распределения аргумент `Arg4` устанавливается в значение `False`
`Ztest`	`Double`	`Arg1, Arg2 As Double, [Arg3]`	Возвращает двустороннее P-значение z-теста. Аргумент `Arg1` содержит массив или диапазон данных, которые тестируются относительно аргумента `Arg2`. Аргумент `Arg2` — тестируемое значение. Аргумент `Arg3` — стандартное отклонение для общего множества данных

Пример: объект WorksheetFunction

```
Sub GetBiggest()
   Dim oWSF As WorksheetFunction
   Dim vaArray As Variant
   Set oWSF = Application.WorksheetFunction
   vaArray = Array(10, 20, 13, 15, 56, 12, 8, 45)
   MsgBox " Наибольшее значение равно " & oWSF.Max(vaArray)
End Sub
```

Объект XmlDataBinding

Представляет подключение к источнику данных для карты XML.

Имя	*Тип возвращаемого значения*	*Описание*
`Application`	`Object`	Возвращает ссылку на приложение Microsoft Excel
`Creator`	`Long`	Возвращает длинное целое число, которое указывает на приложение, создавшее объект

Имя	*Тип возвращаемого значения*	*Описание*
Parent	Object	Возвращает ссылку на родительский объект
SourceUrl	String	Возвращает строку, представляющую содержимое файла данных XML

Методы объекта XmlDataBinding

Имя	*Тип возвращаемого значения*	*Параметры*	*Описание*
ClearSettings	Boolean		Очищает все параметры текущего объекта
LoadSettings	Boolean		Загружает набор параметров
Refresh			Обновляет все данные

Объект XmlMap

Объект XMLMap описывает карту XML, которая добавляется к книге.

Свойства объекта XmlMap

Имя	*Тип возвращаемого значения*	*Описание*
AdjustColumnWidth	Boolean	Устанавливается в значение True для автоматического изменения ширины столбцов при обновлении данных
AppendOnImport	Boolean	Устанавливается в значение True для добавления импортируемых данных к уже существующим. В противном случае импортируемые данные перезаписывают существующие
Application	Long	Возвращает ссылку на приложение Excel
Creator	Long	Указывает на объект, создавший текущий объект
DataBinding	XmlDataBinding	Возвращает объект XmlDataBinding, описывающий привязку для указанной карты схемы
IsExportable	Boolean	Определяет возможность экспорта текущих данных
Name	String	Возвращает имя текущего объекта
Parent	Object	Возвращает ссылку на родительский объект данного объекта
PreserveColumnFilter	Boolean	Определяет необходимость сохранения фильтров столбцов
PreserveNumberFormatting	Boolean	Определяет необходимость сохранения форматов чисел
RootElementName	String	Возвращает строку, описывающую имя корневого элемента в дереве XML
RootElementNamespace	XmlNamespace	Возвращает объект XmlNamespace, соответствующий пространству имен текущего корневого элемента
SaveDataSourceDefinition	Boolean	Определяет необходимость сохранения информации об источнике данных

Имя	*Тип возвращаемого значения*	*Описание*
Schemas	XmlSchemas	Возвращает коллекцию объектов XmlSchema, использующихся в текущей книге
ShowImportExportValidationErrors	Boolean	Устанавливается во время импорта и экспорта в значение True для вывода всех ошибок проверки действительности документа

Методы объекта XmlMap

Имя	*Тип возвращаемого значения*	*Параметры*	*Описание*
Delete			Удаляет текущий объект XmlMap
Export			Экспортирует текущий объект XmlMap
ExportXml			Экспортирует текущий объект XmlMap в виде кода XML, который можно сохранить в файле
Import			Импортирует объект XmlMap
ImportXml			Импортирует объект XmlMap из кода XML

Б

Объектная модель VBE

Объектная модель VBE официально называется “Microsoft Visual Basic for Applications Extensibility 5.3”. Эта библиотека предоставляет доступ к коду и формам внутри приложения, а также различным объектам, составляющим интегрированную среду разработки Visual Basic (Visual Basic Integrated Development Environment – VBIDE). По умолчанию данная объектная библиотека отсутствует в списке ссылок нового проекта. Для использования рассмотренных здесь объектов в приложение необходимо добавить ссылку на библиотеку Microsoft Visual Basic for Applications Extensibility 5.3. Для этого можно воспользоваться командой меню Tools⇨References (Сервис⇨Ссылки) в редакторе VBE. Опыт программирования для Office XP может оказаться полезным при программировании в Office 2003, так как версия библиотеки не изменилась.

Многие объекты из объектной модели VBE имеют те же имена, что и объекты из объектной модели Excel. Для явного выбора объектов из объектной модели VBE необходимо использовать имя VBIDE во всех операторах `Dim`:

```
Dim oWinVB As VBIDE.Window ' Всегда предоставляет объект VBE Window
Dim oWinXL As Excel.Window ' Всегда предоставляет объект Excel Window
Dim oWin As Window         ' Возвращает окно Excel Window
```

Все приложения из поставки Office 2003 применяют одну и ту же среду разработки – редактор VBE. Код и формы, принадлежащие книге Excel, документу Word, базе данных Access, презентации PowerPoint (то есть, “документу”), группируются в проекты Visual Basic (объект `VBProject`). Для каждого документа существует свой проект. Приложения FrontPage и Outlook имеют собственные проекты, которые “принадлежат” приложению.

Связь между объектной моделью Excel и объектной моделью VBE

Существует несколько свойств объектов Excel, предоставляющих связи с объектной моделью VBE. Точно так же существует несколько свойств в объектной модели VBE, предоставляющих связи с объектной моделью Excel. Эти связи используются во многих примерах кода в этом приложении и в главе 14.

Свойство связи от Excel к VBE	*Результирующий элемент VBE*
`Application.VBE`	Объект `VBE`
`Workbook.VBProject`	Объект `VBProject`
`Workbook.CodeName`	Имя объекта `VBComponent` на уровне книги в проекте `VBProject`. Обычно это объект `"ThisWorkbook"` в англоязычных версиях Excel
`Worksheet.Codename`, `Chart.CodeName`	Имя объекта `VBComponent` на уровне листа в проекте `VBProject`. Обычно это объект `"Sheet1"`, `"Chart1"` в англоязычных версиях Excel

Свойство связи от VBE к Excel	*Результирующий элемент Excel*
`VBProject.FileName`	Полное имя книги, если объект `VBProject` является проектом книги Excel и книга была сохранена
`VBComponent.Properties("Name")`	Имя файла книги, если объект `VBComponent` является элементом уровня книги (например, `"ThisWorkbook"`). Для объектов `VBComponent` уровня листа это свойство возвращает имя листа
`VBComponent.Properties("<другие свойства>")`	Возвращает свойства, связанные с объектом Excel, к которым относится объект `VBComponent`

Общие свойства и методы

Большинство объектов из объектной библиотеки VBE предоставляют ряд общих свойств. Для избежания избыточности эти свойства будут перечислены в описании каждого объекта, но подробная информация приводиться не будет. Обычно, если объекты обладают общими свойствами, они являются потомками одного базового класса. Базовые классы и наследование должны быть известны всем, кто знаком с разработкой на основе объектно-ориентированной парадигмы. Дополнительная информация об объектно-ориентированной разработке приводится на сайте Microsoft Development Network (`http://msdn.microsoft.com`).

Имя	*Тип возвращаемого значения*	*Описание*
`Collection`		Только чтение. Возвращает коллекцию, которой принадлежит объект. Например, объект `Reference` принадлежит коллекции `References`. Объекты, принадлежащие коллекциям, предоставляют свойство `Collection`. Коллекции являются средствами группировки одинаковых объектов
`Parent`		Только чтение. Возвращает объект, которому принадлежит коллекция. Например, коллекция `References` принадлежит объекту `VBProject`. Объекты, не принадлежащие коллекциям, предоставляют свойство `Parent`
`VBE`	`VBE`	Только чтение. Возвращает объект редактора Visual Basic Editor, аналогичный объекту `Application` в объектной модели Excel

> Коллекции можно воспринимать как средства группировки похожих объектов. Например, если необходимо хранить информацию из телефонного справочника, можно создать класс объектов, содержащих всю контактную информацию для одного человека. После этого коллекцию можно использовать для группировки всех записей с контактной информацией. Коллекция позволяет применять цикл `For Each` для перебора в ней всех записей.

Большинство объектов из объектной модели VBE хранятся в соответствующих коллекциях. Обычно название коллекции представляет собой множественное число от названия объекта в коллекции. Для простоты каждый объект и связанная с ним коллекция будут рассматриваться в одном и том же разделе. Общие свойства и методы коллекций совпадают со свойствами и методами из коллекции Excel и характеризуются в приложении А. Для каждого объекта будут описываться только уникальные свойства, методы или события.

Объект AddIn и коллекция AddIns

Не стоит путать этот объект с объектом `Addin` из объектной модели Excel. Надстройки VBE представляют собой динамически загружаемые библиотеки, построенные в соответствии с архитектурой Microsoft Component Object Model. Надстройки VBE более известны, как "надстройки COM". Обычно такие надстройки создаются с помощью C++, Visual Basic или любого из языков, основанных на инфраструктуре .NET Framework. Установленные надстройки перечислены в списке в окне VBE Add-Ins. Для загрузки и выгрузки надстроек VBE можно воспользоваться командой меню Add-Ins⇨Add-In Manager (Надстройки⇨Диспетчер надстроек).

Некоторые надстройки VBE используются в качестве инструментов для генерации кода, средств просмотра API с генерацией операторов `Declare` для методов Win32 API и т.д. Для редактора VBE созданы тысячи надстроек, многие из которых можно найти на сайте компании Microsoft.

Общие свойства объекта AddIn

Определения свойств `Collection` и `VBE` приводились в начале приложения.

Свойства объекта AddIn

Имя	*Тип возвращаемого значения*	*Описание*
`Connect`	`Boolean`	Сообщает, активизирована ли надстройка COM. Для запуска надстройки свойство устанавливается в значение `True`. Это свойство похоже на свойство `Installed` объекта Excel `Addin`
`Description`	`String`	Текстовая строка, которая отображается в поле Description (Описание) в окне VBE Addin Manager
`Guid`	`String`	Только чтение. Возвращает глобально уникальный идентификатор надстройки, создаваемый Excel/Visual Basic при компиляции надстройки. Глобально уникальный идентификатор генерируется на основе нескольких параметров. Генерация двух одинаковых глобально уникальных идентификаторов на одном компьютере практически невозможна
`Object`		Для предоставления объекта в редактор VBE используется метод `OnConnection` (обычно предоставляется корневой класс собственной объектной модели, если такая модель существует). После этого для доступа к объекту используется свойство `Object`, посредством которого можно получить доступ к остальной объектной модели надстройки. На данный момент свойство `Object` большинством надстроек не предоставляется
`ProgId`	`String`	Только чтение. Возвращает программный идентификатор надстройки, который состоит из имени проекта надстройки и имени класса подключения (обычно конструктора подключения). Например, если создается проект надстройки `MyAddin` и класс `Designer` для надстройки называется `dsrMyConnection`, программный идентификатор будет выглядеть, как `"MyAddin.dsrMyConnection"`

Методы коллекции AddIns

Имя	*Тип возвращаемого значения*	*Описание*
`Item`	`Addin`	Только чтение. Параметры: `Item As Variant`. Возвращает надстройку. В качестве параметра используется номер или программный идентификатор надстройки (например `MyAddin.dsrMyConnection`)
`Update`		Метод. Обновляет список доступных надстроек COM на основе данных из системного реестра. Используется только при автоматической компиляции надстройки (например, с помощью метода `VBProject.MakeCompiledFile`)

Примеры использования надстроек

В следующем примере код перебирает надстройки, зарегистрированные для использования в редакторе VBE, и выводит информацию об активных надстройках.

```
Sub ListRunningAddins()
   ' Определить надстройку VBE, а не надстройку Excel
   Dim oAddin As VBIDE.Addin

   ' Перебрать надстройку для редактора VBE
   For Each oAddin In Application.VBE.AddIns

      ' Надстройка активизирована (т.е. подключена)?
      If oAddin.Connect Then

         ' Да, поэтому показать идентификатор и описание
         Debug.Print oAddin.ProgId, oAddin.Description
       End If
   Next
End Sub
```

Обратите внимание, что надстройки для редактора VBE не предоставляют свойства для определения имени надстройки.

Объект CodeModule

Объект `CodeModule` содержит весь код для объекта `VBComponent` (например, модуля, диалогового окна `UserForm`, модуля класса или листа Excel). Для каждого компонента может существовать только один объект `CodeModule`. Методы и свойства этого объекта позволяют находить, идентифицировать, модифицировать и добавлять строки кода компонентов проекта. В модуле может существовать больше одной процедуры с одним и тем же именем, если эти процедуры являются процедурами свойств (`Property`).

```
Dim msSelection As String
Property Get TheSelection() As String
   TheSelection = msSelection
End Property

Property Let TheSelection(NewString As String)
   MsSelection = NewString
End Property
```

Следовательно, для уникальной идентификации процедуры необходимо предоставить как ее имя (в данном случае `"TheSelection"`), так и тип интересующей процедуры (vbext_pk_Get для процедуры `Property Get`, vbext_pk_Let для процедуры `Property Let`, vbext_pk_Set для процедуры `Property Set` или vbext_pk_proc для подпрограмм и функций). Функция `ProcOfLine` предоставляет эту информацию для указанного номера строки — имя процедуры возвращается из функции, а тип процедуры возвращается в переменную, предоставляемую в качестве аргумента `ProcKind`. Это одно из нескольких свойств в Office 2002, модифицирующих предоставленные аргументы.

Общие свойства объекта CodeModule

Определения свойств `Parent` и `VBE` приводились в начале приложения.

Свойства объекта CodeModule

Имя	*Тип возвращаемого значения*	*Описание*
CodePane	CodePane	Только чтение. Возвращает активный объект CodePane для модуля. Если видимый объект CodePane не существует, он создается и выводится на экран. Обратите внимание, что объект CodeModule может содержать до двух панелей с кодом, но коллекция CodePanes не создается

CountOf-Declaration-Lines	Long	Только чтение. Возвращает количество строк в начале модуля, содержащих операторы Dim, Type и Option. Если такие строки присутствуют в начале модуля, расположенные за ними комментарии рассматриваются как часть следующей процедуры. Следующий код содержит две строки с объявлениями:
		`Option Explicit`
		`Dim msSelection As String`
		`'Комментарий`
		`Sub ProcedureStart()`
		Если такие операторы отсутствуют, комментарии в начале модуля перед пустой строкой считаются частью объявлений. Следующий код содержит только одну строку объявлений:
		`'Комментарий`
		
		`Sub ProcedureStart()`
		Если после комментария немедленно следует определение процедуры, комментарий включается в состав процедуры, поэтому этот код не содержит строк объявления:
		`'Комментарий`
		`Sub ProcedureStart()`

CountOfLines	Long	Только чтение. Возвращает общее количество строк кода в модуле. Продолжение строк считается отдельными строками
Lines	String	Только чтение. Параметры: StartLine As Long, Count As Long. Возвращает блок кода начиная от строки, на которую указывает параметр StartLine. Количество строк указывается в качестве значения параметра Count
Name	String	(Скрытое свойство) Только чтение. Возвращает имя связанного объекта VBComponent
ProcBodyLine	Long	Только чтение. Параметры: ProcName As String, ProcKind As vbext_ProcKind. Возвращает номер строки, на которой расположено начало процедуры, не включая комментарии перед процедурой. Другими словами, свойство возвращает номер строки, в которой расположено ключевое слово Sub, Function или Property
ProcCount-Lines	Long	Только чтение. Параметры: ProcName As String, ProcKind As vbext_ProcKind. Возвращает количество строк, составляющих процедуру, от комментариев перед началом процедуры до оператора End Sub, End Function или End Property

Имя	*Тип возвращаемого значения*	*Описание*
ProcOfLine	String	Только чтение. Параметры Line As Long [in], ProcKind As Long [out]. Возвращает имя процедуры, в пределах которой находится строка. Аргумент ProcKind модифицируется для предоставления информации о типе процедуры (Sub/Function, Property Let, Get или Set). Обычно это свойство вызывается в первую очередь, а полученные в результате вызова имя и тип процедуры используются для вызова других методов
ProStartLine	Long	Только чтение. Параметры: ProcName As String, ProcKind As vbext_ProcKind. Возвращает номер строки, в которой начинается процедура, включая комментарии. Разность ProcBodyLine-ProcStartLine равна количеству строк комментариев перед началом процедуры

Методы объекта CodeModule

Имя	*Тип возвращаемого значения*	*Параметры*	*Описание*
AddFromFile		FileName As String	Читает код из текстового файла и добавляет его в конец модуля кода. Уникальность имен процедур в пределах модуля кода не проверяется
AddFrom-String		String As String	Добавляет код из строки в конец модуля кода
Create-EventProc	Long	EventName As String, ObjectName As String	Создает в модуле пустую процедуру обработки события, автоматически указывая параметры события. Свойство не может использоваться для стандартных модулей, так как они не поддерживают обработку событий. Параметр ObjectName должен содержать имя допустимого для модуля класса объекта. Параметр EventName должен содержать допустимое название события для указанного объекта
DeleteLines		StartLine As Long, Count As Long	Удаляет строки из модуля кода начиная со строки, на которую указывает параметр StartLine. Количество удаляемых строк указывается в качестве значения параметра Count

Имя	*Тип возвращаемого значения*	*Параметры*	*Описание*
Find	Boolean	Target As String, StartLine As Long, StartColumn As Long, EndLine As Long, EndColumn As Long, WholeWord As Boolean, MatchCase As Boolean, PatternSearch As Boolean	Находит строку в модуле кода или раздел модуля кода. Свойство предоставляет ту же функциональность, что и диалоговое окно Find (Найти) в редакторе VBE
InsertLines		Line As Long, String As String	Добавляет код из строки внутрь модуля кода перед строкой, на которую указывает значение параметра Line
ReplaceLine		Line As Long, String As String	Добавляет код из строки внутрь модуля кода, заменяя строку, на которую указывает значение параметра Line

Пример использования объекта CodeModule

В главе 14 приводится несколько примеров использования объекта CodeModule. В следующем примере выполняется идентификация процедуры, внутри которой расположена указанная строка. При этом выводится тип, имя и количество строк в процедуре.

```
Sub WhichProc()
   Dim lLine As Long, iProcKind As Long, lLineCount As Long
   Dim sProc As String, sMsg As String
   Dim oActiveCM As VBIDE.CodeModule

   lLine = CLng(InputBox("Which line?"))

   ' Отменена?
   If lLine = 0 Then Exit Sub

   ' Получить активный модуль кода
   Set oActiveCM = Application.VBE.ActiveCodePane.CodeModule

   ' Получить имя и тип процедуры в указанной строке
   ' При этом меняется значение параметра iProcKind
   sProc = oActiveCM.ProcOfLine(lLine, iProcKind)

   If sProc = "" Then
      ' Имя процедуры не предоставлено, а значит строка расположена
      ' в разделе объявлений
      sMsg = "Строка расположена в разделе объявлений"
      lLineCount = oActiveCM.CountOfDeclarationLines
   Else
      sMsg = "Строка расположена в "

      ' Показать тип процедуры...
      Select Case iProcKind
         Case vbext_pk_Proc
            sMsg = sMsg & "Подпрограмме Sub или функции Function"
         Case vbext_pk_Get
            sMsg = sMsg & "Процедуре Property Get"
```

```
        Case vbext_pk_Let
           sMsg = sMsg & "Процедуре Property Let"
        Case vbext_pk_Set
           sMsg = sMsg & "Процедуре Property Set"
     End Select

     '... название ...
     sMsg = sMsg & " '" & sProc & "'"

     '...  количество строк в процедуре.
     lLineCount = oActiveCM.ProcCountLines(sProc, iProcKind)
  End If
    ' Вывести сообщение
  MsgBox sMsg & vbCrLf & " состоит из " & lLineCount & " строк."
End Sub
```

Объект CodePane и коллекция CodePanes

Объект CodePane является представлением объекта CodeModule, предоставляющим доступ к уровню интерактивности между разработчиком и редактируемым кодом. Большинство надстроек VBE используют этот код для идентификации текущего объекта CodePane и редактируемой в данный момент строки. После этого код в этой строке модифицируется с помощью методов и свойств объекта CodeModule. Обратите внимание, что для одного объекта CodeModule может существовать больше одного объекта CodePane (например, при разделении окна кода на две панели с помощью горизонтального разделителя).

Общие свойства объекта CodePane

Определения свойств Collection и VBE приводились в начале приложения.

Свойства объекта CodePane

Имя	*Тип возвращаемого значения*	*Описание*
CodeModule	CodeModule	Только чтение. Возвращает объект CodeModule, который содержит код, показанный в объекте CodePane
CodePaneView	vbext_CodePaneview	Только чтение. Возвращает необходимость вывода с помощью объекта CodePane одной процедуры или целого модуля с линиями разделителей между процедурами
CoutOfVisibleLines	Long	Только чтение. Возвращает количество строк, видимых в объекте CodePane. Это свойство и свойство TopLine может использоваться для центрирования строк в окне объекта CodePane (дополнительная информация приводится в следующем примере)
TopLine	Long	Возвращает номер первой строки объекта CodePane в объекте CodeModule
Window	Window	Только чтение. Возвращает объект Window, в котором расположен объект или объекты CodePane

Методы CodePane

Имя	Параметры	Описание

GetSelection	StartLine As Long, StartColumn As Long EndLine As Long, EndColumn As Long	Используется для получения выделенного текста. Все аргументы передаются по ссылке и модифицируются внутри процедуры для предоставления выделения. Все аргументы являются обязательными, но они необходимы только для передачи аргументов для извлекаемых элементов. Например, для получения только первой строки можно воспользоваться:
		`Dim lStart As Long`
		`Application.VBE.ActiveCodePane.Get Selection - lStart, 0, 0, 0`

SetSelection	StartLine As Long, StartColumn As Long, EndLine As Long, EndColumn As Long	Используется для установки положения выделенного текста. Обычно программы получают текущее выделение с помощью метода GetSelection, модифицируют выделение и устанавливают выделенный фрагмент обратно с помощью метода SetSelection. Дополнительная информация по использованию этого свойства приводится в главе 23 в разделе, описывающем процедуру PrintProcedure
Show		Открывает и выводит на экран объект CodePane. При этом объект активизируется

Свойства коллекции CodePanes

Коллекция CodePanes содержит все открытые объекты CodePane из редактора VBE.

Имя	Тип возвращаемого значения	Описание
Current	CodePane	Только чтение. Возвращает активный объект CodePane. Совпадает со свойством Application.VBE.ActiveCodePane

Примеры использования объекта CodePane

В главе 23 приводилось несколько примеров использования объекта CodePane. В следующем примере идентифицируется текущее выделение и выполняется центрирование окна CodePane.

```
Sub CenterSelectionInWindow()
   Dim oCP As VBIDE.CodePane
   Dim lStartLine As Long, lEndLine As Long
     Dim lVisibleLines As Long, lNewTop As Long

   ' Получить активный объект CodePane
   Set oCP = Application.VBE.ActiveCodePane

   ' С помощью объекта CodePane ...
   With oCP
      ' Получить первую и последнюю строки выделения
      .GetSelection lStartLine, 0, lEndLine, 0

      ' Сколько строк помещается в окне?
      lVisibleLines = .CountOfVisibleLines
```

```
    ' Какая строка должна быть первой?
    lNewTop = (lStartLine + lEndLine - lVisibleLines)\2

    ' Настроить окно для вывода кода из этой строки
    .TopLine = lNewTop
  End With
End Sub
```

Объект CommandBarEvents

В пределах редактора VBE свойство OnAction кнопки командной панели не играет никакой роли. Подпрограмма, на которую указывает это свойство, не запускается при щелчке на кнопке. Вместо этого объектная модель VBE предоставляет объект CommandBarEvents, который подключается к любой кнопке, включая встроенные и собственные элементы. При этом объект вызывает процедуры обработки выбранных событий. В пакете Office 2002 этот объект обеспечивает обработку только события Click. Следовательно, объект предоставляет функциональность свойства OnAction. Основным отличием является наличие аргументов, позволяющих модифицировать поведение события Click. Кроме этого, объект CommandBarEvents предоставляет расширенный интерфейс, тем самым позволяя компании Microsoft реализовывать более богатую модель событий в будущих версиях редактора VBE (например, BeforePupUp, BeforeRightClick и стандартные события мыши).

События объекта CommandBarEvents

Имя	*Параметры*	*Описание*
Click	CommandBarControl As Object, handled As Boolean, CancelDefault As Boolean	Возникает при щелчке на связанной кнопке командной панели. Параметр CommandBarControl передается в процедуру обработки события в качестве параметра. К одному элементу управления можно подключиться с помощью нескольких объектов CommandBarControl. События возникают в обратном порядке (событие, настроенное позднее, возникает первым). Процедура обработки события может устанавливать параметр Handled в значение True, сообщая последующим обработчикам, что событие уже обработано. Кроме этого, объект CommandBarControl используется для подключения к встроенным элементам меню. Если событие обрабатывается кодом, параметр CancelDefault должен быть установлен в значение True. Это приведет к отмене нормальной операции обработки события пункта меню

Примеры использования объекта CommandBarEvents

Добавьте следующий код в модуль класса CBarEvents.

```
Public WithEvents oCBEvents As VBIDE.CommandBarEvents

' Подключиться к событию Click пункта меню
Private Sub oCBEvents_Click(ByVal CommandBarControl As Object, _
        handled As Boolean, CancelDefault As Boolean)

   Debug.Print "Щелчок на " & CommandBarControl.Caption
End Sub
```

Следующий код добавьте в обычный модуль.

```
' Определить коллекцию, в которой будут храниться все
' экземпляры класса событий
Dim ocolMenus As New Collection

Sub AddMenus()

   ' Объявить элементы управления  CommandBar
   Dim oBar As CommandBar
   Dim oBtn1 As CommandBarButton, oBtn2 As CommandBarButton

   ' Добавить объект, содержащий экземпляр класса событий
   Dim oCBE As CBarEvents

   ' Получить панель меню VBE
   Set oBar = Application.VBE.CommandBars("Menu Bar")

   ' Добавить пункт в панель меню
   Set oBtn1 = oBar.Controls.Add(Type:=msoControlButton, temporary:=True)
   oBtn1.Caption = "Menu1"
   oBtn1.Style = msoButtonCaption

   ' Создать новый экземпляр обработчика CommandBarEvent
   Set oCBE = New CBarEvents

   ' Связать обработчик CommandBarEvent с созданным пунктом меню
   Set oCBE.oCBEvents = Application.VBE.Events.CommandBarEvents(oBtn1)

   ' И добавить экземпляр обработчика события для коллекции
   ocolMenus.Add oCBE

   ' Повторить для второго меню
   Set oBtn2 = oBar.Controls.Add(Type:=msoControlButton,
➥ temporary:=True)
   oBtn2.Caption = "Menu2"
   oBtn2.Style = msoButtonCaption

   Set oCBE = New CBarEvents
   Set oCBE.oCBEvents = Application.VBE.Events.CommandBarEvents(oBtn2)
   ocolMenus.Add oCBE
End Sub
```

При запуске подпрограммы AddMenus в стандартную панель меню редактора VBE добавляется два новых меню. Для обработки событий Click новых пунктов меню используется класс CommandBarEvents. При щелчке на каждом из пунктов меню в окне Immediate (Проверка) выводится заголовок пункта меню.

Объект Events

Объект Events является контейнером верхнего уровня для модели событий в редакторе VBE. В Office 2002 в этом контейнере хранятся объекты событий, связанные со щелчком на кнопках командных панелей и с добавлением/удалением ссылок. Модель расширения редактора VBE основана на модели расширения Visual Basic, которая содержит больший набор событий.

Свойства объекта Events

Имя	*Тип возвращаемого значения*	*Описание*
CommandBar-Events	CommandBar-Events	Только чтение. Параметры: CommandBarControl. Выполняет необходимое связывание для подключения объекта CommandBarEvents к конкретной кнопке командной панели
References-Events	References-Events	Только чтение. Параметры: VBProject. Выполняет необходимое связывание для подключения объекта ReferencesEvents к конкретному проекту

Примеры использования объекта Events

Примеры использования объекта Events приводятся в разделах, посвященных объектам CommandBarEvents и ReferencesEvents.

Коллекция LinkedWindows

В коллекции LinkedWindows хранятся все связанные окна в рабочем пространстве редактора VBE. Надстройки COM, написанные с помощью Visual Basic 5/6 (но не в Office 2002 Developer Edition), могут добавлять собственные окна в эту коллекцию. В среде Office возможность связывания встроенных окон ограничена. Обратите внимание, что при разрыве связи с последующим связыванием встроенных окон они не возвращаются в исходную позицию.

Методы коллекции LinkedWindows

Имя	*Тип возвращаемого значения*	*Описание*
Add		Метод. Параметры: Window As Window. Связывает указанное окно
Remove		Метод. Параметры: Window As Window. Разрывает связь для указанного окна

Объект Property и коллекция Properties

Все объекты VBComponent в пределах проекта обладают коллекцией Properties. Хранящиеся в коллекции свойства соответствуют элементам, показанным в окне Properties в редакторе VBE. Коллекция Properties содержит многие свойства объектов Excel, соответствующих объектам VBComponent.

Общие свойства объекта Property

Определения свойств Collection, Parent и VBE приводятся в начале приложения.

Свойства объекта Property

Имя	*Тип возвращаемого значения*	*Описание*
IndexedValue	Variant	Параметры: Index1, [Index2], [Index3], [Index4]. Значение свойства Value объекта Property представляет собой массив размером до четырех индексов. Свойство IndexedValue позволяет прочитать один элемент возвращаемого массива
Name	String	Только чтение. Возвращает имя свойства. Кроме этого, используется для обращения к конкретному свойству
NumIndices	Integer	Только чтение. Если значением объекта Property является массив, это свойство возвращает количество индексов (размерностей) массива. Если значение не является массивом, это свойство возвращает значение 0
Object	Object	Свойство Object используется для получения ссылки на объект, извлекаемый из объекта Property
Value	Variant	Значение объекта Property

Свойство Name может легко ввести в заблуждение.

Элемент	*Ссылается на*
Worksheet.CodeName	Кодовое имя объекта VBComponent (только чтение)
VBComponent.NAme	Кодовое имя объекта VBComponent (чтение/запись)
VBComponent.Properties("CodeName")	Кодовое имя объекта VBComponent (только чтение)
VBComponent.Properties("_CodeName")	Кодовое имя объекта VBComponent (чтение/запись)*
VBComponent.Properties("Name")	Имя листа (чтение/запись)
VBComponent.Properties("Name").Name	"Name"

Примеры использования объекта Property

В этом примере идентифицируется лист, содержащий указанный объект VBComponent.

```
Sub IdentifyWorkbook()
   Dim oBk As Workbook

   ' Получить книгу, содержащую указанный объект VBComponent
   Set oBk = Application.VBE.ActiveVBProject.VBComponents("Sheet1").
↳Properties("Parent")
.Object

   MsgBox oBk.Name
End Sub
```

* *Это единственный надежный способ изменения кодового имени листа в Excel 97.*

Объект Reference и коллекция References

Объект Reference связывает проект VBProject и внешний файл, который может быть объектной библиотекой (например, при связывании с объектной библиотекой Word), элементом управления (например, Windows Common Controls), динамически подключаемой библиотекой ActiveX или еще одним проектом VBProject. Создание ссылки на внешний объкт позволяет осуществить раннюю привязку. Это значит, что объект, на который ссылается проект, будет находиться в той же области памяти, все связи будут проверяться на этапе компиляции, и Excel предоставит всплывающие подсказки при программировании с использованием связанных объектов.

При запуске приложения на другом компьютере оно может не обнаружить всех необходимых объектов. Объект Reference и коллекция References предоставляют доступ к таким ссылкам, позволяя проверить существование всех необходимых файлов перед их использованием.

Общие свойства объекта Reference

Определения свойств Collection и VBE приводились в начале приложения.

Свойства объекта Reference

Имя	*Тип возвращаемого значения*	*Описание*
BuiltIn	Boolean	Только чтение. Указывает, является ли ссылка встроенной или она была добавлена разработчиком. Ссылки на библиотеки "Visual Basic for Applications" и "Microsoft Excel 10.0 Object Library" являются встроенными и не могут быть удалены
Description	String	Только чтение. Возвращает описание ссылки, текст которого приводится в окне Object Browser (Просмотр объектов)
FullPath	String	Только чтение. Возвращает путь к книге, файлу DLL, файлу OCX, файлу TLB или файлу OLB, на который указывает ссылка
Guid	String	Только чтение. Возвращает глобально уникальный идентификатор ссылки
IsBroken	Boolean	Только чтение. Возвращает значение True, если ссылка недействительна (соответствующий файл отсутствует на компьютере)
Major	Long	Только чтение. Возвращает номер основной версии файла
Minor	Long	Только чтение. Возвращает номер дополнительной версии файла
Name	String	Только чтение. Возвращает короткое имя ссылки (например, "VBA" или "Excel")
Type	vbext_RefKind	Только чтение. Возвращает тип ссылки. vbext_rk_TypeLib для библиотек DLL, vbext_rk_Project для других проектов VBProject

Методы коллекции References

Имя	*Тип возвращаемого значения*	*Описание*
AddFromFile	Reference	Метод. Параметры: `FileName As String`. Добавляет в `VBProject` ссылку на конкретный файл. Этот метод используется только для создания ссылок на книги
AddFromGuid	Reference	Метод. Параметры: `Guid As String`, `Major As Long`, `Minor As Long`. Добавляет в `VBProject` ссылку на конкретную библиотеку DLL, библиотеку типов и т.д. Имя файла библиотеки, расположение и номер версии могут измениться со временем, но постоянство глобально уникального идентификатора гарантировано. Следовательно, глобально уникальный идентификатор должен использоваться при добавлении ссылки на библиотеку DLL, библиотеку типов и т.д. Если интересует конкретная версия библиотеки DLL, можно запросить основной и дополнительный номера версий
Remove		Метод. Параметры: `Reference As Reference`. Удаляет ссылку из проекта `VBProject`

События коллекции References

Коллекция `References` предоставляет два события, используемые для обнаружения добавления и удаления элементов коллекции. Например, данные события можно применять для создания диалогового окна "Десятка популярных ссылок". Для этого события объекта `Application` используются для обнаружения открытия или создания книги, а события коллекции References из объекта `VBProject` — для обнаружения добавления конкретной ссылки в проект. Список таких ссылок можно сохранить и выводить в диалоговом окне, похожем на диалоговое окно Tools⇨References (Сервис⇨Ссылки) в редакторе VBE (но без ненужной дополнительной информации).

Имя	*Параметры*	*Описание*
ItemAdded	`Reference As VBIDE.Reference`	Возникает при добавлении ссылки в интересующий проект
ItemRemoved	`Reference As VBIDE.Reference`	Возникает при удалении ссылки из интересующего проекта

Примеры использования объекта Reference

В этом примере выполняется проверка отсутствующих ссылок. При наличии таких ссылок выводится сообщение для пользователя.

```
Function HasMissingRefs() As Boolean
   Dim oRef As VBIDE.Reference

   ' Перебрать все ссылки в проекте
   For Each oRef In ThisWorkbook.VBProject.References

      ' Ссылка указывает на отсутствующий файл?
      If oRef.IsBroken Then

         ' Да - выводить разные сообщения для  ссылок на книги и
```

```
        ' библиотеки DLL
        If oRef.Type = vbext_rk_Project Then
           MsgBox "Невозможно найти книгу " & oRef.FullPath & _
                  ", которая необходима для работы приложения."
        Else
           MsgBox "Это приложение требует доступа к объектной
↳ библиотеке '" & _
                  oRef.Description & "', которая не установлена
↳ в системе."
        End If

        ' Указать, что некоторые ссылки указывают на отсутствующие
        ' файлы или библиотеки
        HasMissingRefs = True
     End If
  Next
End Function
```

В следующем примере показан основной код слежения за добавлением или удалением ссылок в проекты (этот код может использоваться для создания списка 10 популярных ссылок). Реализация кода состоит из четырех этапов:

- вначале необходимо подключиться к событиям коллекции References объектов VBProject во всех открытых книгах;
- далее подключиться к событиям объекта Application для обнаружения открытия или создания книги;
- при создании или открытии книги подключиться к событиям коллекции References объекта VBProject;
- при возникновении события References выполнить соответствующую операцию (в данном случае просто вывести сообщение).

Этот код добавляется в модуль класса CRefEvents и реализует четвертый этап.

```
' Объект WithEvents позволяет подключится к событиям коллекции References
Public WithEvents oRefEvt As VBIDE.References

' Кроме этого, необходимо сохранить ссылку на отслеживаемую книгу
Public oWorkbook As Workbook

' В книгу была добавлена ссылка
Private Sub oRefEvt_ItemAdded(ByVal Reference As VBIDE.Reference)
   Debug.Print "Добавлена ссылка '" & _
               Reference.Description & _
               "' в книгу " & oWorkbook.Name
End Sub

' Из книги удаляется ссылка
Private Sub oRefEvt_ItemRemoved(ByVal Reference As VBIDE.Reference)
   Debug.Print "Удалена ссылка '" & _
               Reference.Description & _
               "' из книги " & oWorkbook.Name
End Sub
```

Следующий код добавляется в модуль класса CAppEvents и реализует третий этап.

```
' Объект WithEvents позволяет подключится к событиям объекта Application
Public WithEvents oApp As Application

' Подключиться к событиям коллекции References новой книги
```

```
Private Sub oApp_NewWorkbook(ByVal Wb As Workbook)

    ' Новый экземпляр обработчика событий для коллекции References
    Dim oRefEvents As New CRefEvents

    ' Указать, к какой книге подключается обработчик
    Set oRefEvents.oWorkbook = Wb

    ' Предоставить объект References
    Set oRefEvents.oRefEvt = Wb.VBProject.References

    ' Добавить обработчик событий в коллекцию обработчиков
    oRefHooks.Add oRefEvents
End Sub

' Подключится к событиям коллекции References открытых книг
Private Sub oApp_WorkbookOpen(ByVal Wb As Workbook)
    Dim oRefEvents As New CRefEvents

    Set oRefEvents.oWorkbook = Wb
    Set oRefEvents.oRefEvt = Wb.VBProject.References
    oRefHooks.Add oRefEvents
End Sub
```

Данный код добавляется в нормальный модуль и реализует первый и второй этапы.

```
' Один экземпляр подключения к событиям объекта Application
Public oAppHooks As New CAppEvents

' Большое количество экземпляров подключений к событиям
' коллекции References
Public oRefHooks As New Collection

Sub SetUpReferenceHooking()
    Dim oRefEvents As CRefEvents
    Dim oBk As Workbook

    Set oRefHooks = Nothing

    ' Этап 1: Перебрать все существующие проекты и подключится
    ' к событиям ссылок
    For Each oBk In Workbooks
        If Not oBk Is ThisWorkbook Then
            Set oRefEvents = New CRefEvents
            Set oRefEvents.oWorkbook = oBk
            Set oRefEvents.oRefEvt = oBk.VBProject.References
            oRefHooks.Add oRefEvents
        End If
    Next
    ' Этап 2: Подключиться к событиям объекта Application и следить
    ' за добавлением новых проектов
    Set oAppHooks.oApp = Application
End Sub
```

Объект ReferencesEvents

Так же, как объект `CommandBarEvents` предоставляет событие `Click` для командной панели, объект `ReferencesEvents` предоставляет два события для коллекции `References` из объекта `VBProject`. Объект `ReferencesEvents` может показаться избыточным. Все обрабатываемые им события предоставляются коллекцией `References`

из объекта VBProject. Единственным отличием (кроме определения) является работа объекта ReferencesEvents с объектом VBProject, а не с коллекцией References из объекта VBProject. Обратите внимание, что проект VBProject компилируется при добавлении или удалении объекта Reference, что приводит к потере всех переменных и экземпляров классов. Следовательно, объект VBProject не может контролировать собственные события для коллекции References.

События объекта ReferencesEvents

Имя	*Тип возвращаемого значения*	*Описание*
ItemAdded	Reference As VBIDE.Reference	Возникает при добавлении объекта Reference в контролируемый проект VBProject
ItemRemoved	Reference As VBIDE.Reference	Возникает при удалении объекта Reference из контролируемого проекта VBProject

Примеры использования объекта ReferencesEvents

Примеры использования объекта ReferencesEvents не отличаются от примеров использования событий коллекции References. Единственное отличие заключается в способе обработки событий (этот код выделен полужирным шрифтом).

В класс CRefEvents необходимо добавить следующий код:

```
' Объект WithEvents позволяет подключится к событиям
' коллекции References
Public WithEvents oRefEvt As VBIDE.References Events
```

В класс CAppEvents необходимо добавить такой код:

```
' Подключится к событиям коллекции References открытых книг
Private Sub oApp_WorkbookOpen(ByVal Wb As Workbook)
   Dim oRefEvents As New CRefEvents

   Set oRefEvents.oWorkbook = Wb
   Set oRefEvents.oRefEvt = _
      Application.VBE.Events.ReferencesEvents(Wb.VBProject)
   oRefHooks.Add oRefEvents
End Sub
```

Следующий код необходимо добавить в нормальный класс:

```
Set oRefEvents.oWorkbook = oBk
Set oRefEvents.oRefEvt = _
  Application.VBE.Events.ReferencesEvents(oBk.VBProject)
```

Объект VBComponent и коллекция VBComponents

В коллекции VBComponents хранятся все модули, модули классов (включая код листов) и диалоговые окна UserForm из проекта VBProject. Все эти компоненты являются различными типами объекта VBComponent. Каждый объект VBComponent хранит код в объекте CodeModule, а некоторые объекты VBComponent (например, диалоговые окна UserForm)

обладают графическим интерфейсом, который называется конструктором (Designer). Конструктор предоставляет возможность модификации графических элементов объекта VBComponent, например, добавления элементов управления в диалоговое окно UserForm.

Общие свойства объекта VBComponent

Определения свойств Collection и VBE приводились в начале приложения.

Свойства объекта VBComponent

Имя	*Тип возвращаемого значения*	*Описание*
CodeModule	CodeModule	Только чтение. Возвращает объект CodeModule, в котором хранится код VBA
Designer		Только чтение. Возвращает объект Designer, предоставляющий доступ к графическим элементам на этапе проектирования
DesignerID	String	Только чтение. Возвращает идентификатор объекта Designer, что позволяет определить тип конструктора. Например, идентификатор конструктора для диалогового окна UserForm равен Forms.Form. Идентификатор конструктора для AddinConnection в Office 2002 Developer Edition равен MSAddnDr.AddInDesigner
Designer-Window	Window	Только чтение. Возвращает объект Window, представляющий окно конструктора. (В окне Object Browser (Просмотр объектов) это свойство показано как метод, так как оно открывает окно Window, если оно еще не открыто)
HasOpen-Designer	Boolean	Только чтение. Указывает, открыт ли конструктор компонента
Name	String	Имя объекта VBComponent
Properties	Properties	Только чтение. Возвращает коллекцию Properties, которая предоставляет доступ к свойствам из окна Property (Свойства) и свойствам связанных объектов Excel, если объект VBComponent содержит код для объекта Excel. Дополнительная информация приводится в описании объекта Property
Saved	Boolean	Только чтение. Возвращает присутствие несохраненных изменений в объекте VBComponent. Это свойство аналогично свойству Saved книги Excel, но применяется к каждому объекту VBComponent по отдельности
Type	vbext_Component-Type	Только чтение. Возвращает тип объекта VBComponent. vbext_ct_StdModule — нормальный модуль vbext_ct_ClassModule — модуль класса vbext_ct_MSForm — диалоговое окно UserForm vbext_ct_Document — объект Excel vbext_ct_ActiveXDesigner — все остальные типы

Методы объекта VBComponent

Имя	Параметры	Описание
Activate		Отображает основное окно (модуль кода или конструктор) объекта VBComponent и делает его активным
Export	FileName As String	Сохраняет объект VBComponent в файле отдельно от книги

Методы коллекции VBComponent

Имя	Тип возвращаемого значения	Описание
Add	VBComponent	Параметры: ComponentType. Добавляет в проект новый встроенный объект VBComponent. Параметр ComponentType может иметь значение vbext_ct_StdModule, vbext_ct_ClassModule или vbext_ct_MSForm
AddCustom	VBComponent	Параметры: ProgId. Добавляет в проект новый невстроенный объект. В результате всегда возвращается тип vbext_ct_ActiveXDesigner. Похоже, что невстроенные компоненты Visual Basic могут добавляться только в проекты ActiveX DLL, но не в проекты книг Excel
Import	VBComponent	Параметры: FileName. Добавляет в проект новый объект VBComponent из файла (обычно это экспортированный ранее объект VBComponent)
Remove		Параметры: VBComponent. Удаляет объект VBComponent из проекта

Примеры использования объекта VBComponent

Объект VBComponent и его свойства и методы используются во многих примерах в этом разделе и в главе 23. Следующий код экспортирует диалоговое окно UserForm из содержащей код книги, импортирует диалоговое окно в новую книгу и переименовывает диалоговое окно. После этого код добавляет стандартный модуль, код для вывода диалогового окна и вызывает подпрограмму, выводящую диалоговое окно в новой книге.

```
Sub CopyAndShowUserForm()
   Dim oNewBk As Workbook, oVBC As VBIDE.VBComponent

   ' Создать новую книгу
   Set oNewBk = Workbooks.Add

   ' Экспортировать диалоговое окно UserForm из этой книги на диск
   ThisWorkbook.VBProject.VBComponents("UserForm1").Export "c:\temp.frm"

   ' Импортировать диалоговое окно UserForm в новую книгу
   Set oVBC = oNewBk.VBProject.VBComponents.Import("c:\temp.frm")

   ' Переименовать диалоговое окно UserForm
   oVBC.Name = "MyForm"

   ' Добавить стандартный модуль в новую книгу
   Set oVBC = oNewBk.VBProject.VBComponents.Add(vbext_ct_StdModule)
```

```
    ' Добавить код отображения диалогового окна в стандартный модуль
    oVBC.CodeModule.AddFromString _
            "Sub ShowMyForm()" & vbCrLf & _
            "    MyForm.Show" & vbCrLf & _
            "End Sub" & vbCrLf

    ' Закрыть панель кода, которую Excel открыл при добавлении
    ' кода в модуль
    oVBC.CodeModule.CodePane.Window.Close

    ' Удалить экспортированный файл
    Kill "c:\temp.frm"

    ' Вызвать новую подпрограмму для отображения импортированного
    ' диалогового окна UserForm
    Application.Run oNewBk.Name & "!ShowMyForm"
End Sub
```

Объект VBE

Объект VBE находится на вершине объектной библиотеки VBIDE и является аналогом объекта `Application` из объектной библиотеки Excel. Этот объект выступает в роли контейнера для командных панелей, надстроек, окон и других компонентов среды разработки Visual Basic. Кроме этого, объект VBE предоставляет информацию об объектах, которые в данный момент модифицируются пользователем. К сожалению, этот объект не предоставляет доступ к параметрам среды разработки VBIDE (параметры кода, форматы редактирования, обработка ошибок) и к событиям редактирования (например, выделение различных проектов, добавление или удаление строк кода и т.д.).

Свойства объекта VBE

Имя	*Тип возвращаемого значения*	*Описание*
`ActiveCodePane`	`CodePane`	Устанавливает/возвращает объект `CodePane`, который в данный момент редактируется пользователем. Обычно это свойство используется для идентификации обрабатываемого объекта или для принудительного переключения пользователя на обработку конкретного объекта `CodePane`
`ActiveVBProject`	`VBProject`	Устанавливает/возвращает выделенный в окне Project Explorer (Окно проекта) объект `VBProject`. Если в окне выделен объект `VBComponent`, то это свойство возвращает объект `VBProject`, содержащий выделенный объект
`ActiveWindow`	`Window`	Только чтение. Возвращает активный объект `Window`, который может соответствовать панели кода, конструктору или одному из окон VBIDE (например, окну Project Explorer, окну Immediate и т.д.)
`Addins`	`Addins`	Только чтение. Возвращает коллекцию всех надстроек COM, зарегистрированых для использования в среде разработки VBIDE. Дополнительная информация приводится в описании объекта `AddIn`

Имя	*Тип возвращаемого значения*	*Описание*
`CodePanes`	`CodePanes`	Только чтение. Возвращает коллекцию всех открытых объектов `CodePane` в среде разработки VBIDE. Дополнительная информация приводится в описании объекта `CodePane`
`CommandBars`	`CommandBars`	Только чтение. Возвращает коллекцию всех командных панелей в пределах среды разработки VBIDE
`Events`	`Events`	Только чтение. Возвращает объект, содержащий все события в среде разработки VBIDE. Дополнительная информация приводится в описании объекта `Events`
`MainWindow`	`Window`	Только чтение. Возвращает объект `Window`, представляющий основное окно среды разработки VBIDE
`SelectedVB-Components`	`VBComponent`	Только чтение. Возвращает объект `VBComponent`, выделенный в окне Project Explorer. Обратите внимание, что обычно (но не всегда) значение этого свойства совпадает со значением свойства `ActiveCodePane`
`VBProjects`	`VBProjects`	Только чтение. Возвращает коллекцию всех объектов `VBProject` в пределах среды разработки VBIDE. Это касается как книг Excel, так и библиотек ActiveX DLL
`Version`	`String`	Только чтение. Возвращает номер версии библиотеки Extensibility (для пакета Office 2002 выводится номер версии 6.0)
`Windows`	`Windows`	Только чтение. Возвращает коллекцию всех открытых окон в среде разработки VBIDE. Дополнительная информация приводится в описании объекта `Windows`

Примеры использования объекта VBE

Большинство примеров в этом разделе и в главе 23 используют свойства объекта VBE. Следующая строка кода запускает редактор VBE:

```
Application.VBE.MainWindow.Visible = True
```

Объект VBProject и коллекция VBProjects

Объект `VBProject` представляет весь код в пределах книги, включая код листов, модулей, модулей классов и диалоговых окон `UserForm`. В пакете Office 2002 Developer Edition объект `VBProject` может быть отдельным проектом и поддерживать компиляцию в виде библиотеки ActiveX DLL.

Общие свойства объекта VBProject

Определения свойств `Collection` и `VBE` приводились в начале приложения.

Свойства объекта VBProject

Имя	*Тип возвращаемого значения*	*Описание*
BuildFileName	String	Используется только для библиотек ActiveX DLL. Устанавливает/возвращает имя файла DLL, в который компилируется проект
Description	String	Используется только для библиотек ActiveX DLL. Устанавливает/возвращает описание библиотеки DLL, которое доступно в списке из диалогового окна Tools⇨References (Сервис⇨Ссылки)
FileName	String	Только чтение. Для проектов книг возвращает полное имя книги. Для проектов ActiveX DLL имя файла с исходным кодом проекта (*.vba). Если файл еще не сохранялся, при попытке чтения значения этого свойства возникает ошибка времени выполнения
HelpContextID	Long	Идентифицирует принятый по умолчанию идентификатор контекста в файле справочного руководства для проекта
HelpFile	String	Устанавливает/возвращает файл справочного руководства для проекта. Каждому диалоговому окну UserForm и элементу управления в пределах проекта можно присвоить собственный идентификатор контекста, что позволит получить доступ к соответствующим разделам в файле справочного руководства
Mode	vbext_VBAMode	Только чтение. Возвращает режим работы объекта VBProject (проектирование, выполнение, остановка). Обратите внимание, что разные объекты VBProject могут работать в разных режимах выполнения одновременно (например, проект надстройки ActiveX COM может выполняться в то же время, когда другой проект находится в режиме проектирования)
Name	String	Возвращает имя проекта
Protection	vbext_ProjectProtection	Только чтение. Возвращает необходимость блокировки проекта для просмотра. Заблокированные проекты предоставляют доступ только к объекту VBProject. Любая попытка перемещения на уровни ниже VBProject приводит к появлению сообщения об ошибке. Обратите внимание, что если на объект VBProject установлена защита, но во время пользовательского сеанса она была снята, свойство Protection возвращает значение vbext_pp_none на протяжении всего сеанса
References	References	Только чтение. Возвращает коллекцию объектов Reference для объекта VBProject. Дополнительная информация приводится в описании коллекции References
Saved	Boolean	Только чтение. Указывает на присутствие в объекте VBProject изменений, внесенных после последнего сохранения. Для проектов Excel значение этого свойства совпадает со значением свойства Saved объекта книги

Имя	*Тип возвращаемого значения*	*Описание*
`Type`	`vbext_ProjectType`	Только чтение. Возвращает тип проекта — собственный проект (книга Excel, документ Word, база данных Access) или проект ActiveX DLL
`VBComponents`	`VBComponents`	Только чтение. Возвращает коллекцию `VBComponents` для проекта. Дополнительная информация приводится в описании коллекции `VBComponents`

Методы объекта VBProject

Имя	*Тип возвращаемого значения*	*Параметры*	*Описание*
`MakeCompiledFile`			Только для проектов ActiveX DLL. Компилирует проект и создает файл библиотеки DLL
`SaveAs`		`FileName As String`	Только для проектов ActiveX DLL. Сохраняет проект в файле

Методы коллекции VBProjects

Имя	*Тип возвращаемого значения*	*Описание*
`Add`	`VBProject`	Метод. Параметры: `Type`. Добавляет новый проект в среду разработки VBE. С помощью этого метода можно добавлять только изолированные проекты (библиотеки ActiveX DLL)
`Remove`		Метод. Параметры: `lpc As VBProject`. Удаляет объект `VBProject` из среды разработки VBE. Используется только для проектов ActiveX DLL

Примеры использования объектов VBProject

Большинство примеров в этом разделе используют объект `VBProject` и его свойства. В этом примере перечисляются имена всех объектов `VBComponent` для всех незаблокированных проектов в среде разработки VBE.

```
Sub PrintComponents()
   Dim oVBP As VBIDE.VBProject
   Dim oVBC As VBIDE.VBComponent

   ' Перебрать все проекты в среде разработки VBE
   For Each oVBP In Application.VBE.VBProjects

      ' Если на проект не установлена защита...
      If oVBP.Protection = vbext_pp_none Then

         '... перебрать его компоненты
         For Each oVBC In oVBP.VBComponents
            Debug.Print oVBP.Name & "." & oVBC.Name
         Next
      End If
   Next
End Sub
```

Объект Window и коллекция Windows

Объект Window соответствует одному окну в редакторе VBE, включая основное окно редактора, встроенные окна Project Explorer (Окно проекта), Immediate (Проверка), Debug (Отладка) и окно Watch (Контрольные значения), а также диалоговые окна для объектов CodePane и Designer.

Общие свойства объекта Window

Определения свойств Collection и VBE приводятся в начале приложения.

Свойства объекта Window

Имя	*Тип возвращаемого значения*	*Описание*
Caption	String	Только чтение. Возвращает заголовок окна, показанный в строке заголовка
Height	Long	Устанавливает/возвращает высоту окна в твипах (1 твип = 1/20 пункта). Не оказывает влияния на связанные окна
HWnd	Long	Возвращает дескриптор окна, который используется в вызовах Windows API
Left	Long	Устанавливает/возвращает положение левого края окна на экране в твипах (1 твип = 1/20 пункта). Не оказывает влияния на связанные окна
Linked-WindowFrame	Window	Только чтение. В редакторе VBE поддерживается возможность связывания окон (например, во время присоединения). Это свойство возвращает еще один объект Window, который соответствует фрейму, вмещающему в себя окна. Если окно не является связанным, это свойство возвращает значение Nothing
Linked-Windows	LinkedWindows	Только чтение. Возвращает коллекцию окон, связанных с объектом Window (например, при присоединении окон)
Top	Long	Устанавливает/возвращает положение верхнего края окна на экране в твипах (1 твип = 1/20 пункта). Не оказывает влияния на связанные окна
Type	vbext_WindowType	Только чтение. Возвращает тип окна, например CodePane, окно Immediate, основное окно и т.д.
Visible	Boolean	Устанавливает/возвращает видимость окна на экране
Width	Long	Устанавливает/возвращает ширину окна в твипах (1 твип = 1/20 пункта). Не оказывает влияния на связанные окна
WindowState	vbext_WindowState	Возвращает состояние окна — свернуто, развернуто, нормальное

Методы объекта Window

Имя	*Тип возвращаемого значения*	*Описание*
`Close`		Закрывает окно
`SetFocus`		Открывает и активизирует окно, выводит его на экран и предоставляет фокус

Методы коллекции Windows

Имя	*Тип возвращаемого значения*	*Описание*
`CreateToolWindow`	`Window`	Параметры: `AddInInst`, `ProgId`, `Caption`, `GuidPosition`, `DocObj`. Этот метод используется только при создании надстроек COM в среде разработки Visual Basic 5/6 при создании поддерживающего присоединения окна для редактора VBE

Примеры использования объекта Window

Код из этого примера закрывает все окна кода и окна конструктора в редакторе VBE.

```
Sub CloseAllCodeWindows()
   Dim oWin As VBIDE.Window

   ' Перебрать все открытые окна в редакторе VBE
   For Each oWin In Application.VBE.Windows

      ' Закрыть окно в зависимости от типа
      Select Case oWin.Type
         Case vbext_wt_Browser, vbext_wt_CodeWindow, vbext_wt_Designer

            ' Закрыть окно Object Browser, окна кода и окна
            ' конструктора
            Debug.Print "Закрыто окно '" & oWin.Caption
            oWin.Close

         Case Else
            ' Не закрывать остальные окна
            Debug.Print "Сохраненно окно '" & oWin.Caption &
      End Select
   Next
End Sub
```

В

Объектная модель Office 2003

Общие свойства коллекций и связанных с ними объектов

Большинство объектов из объектной модели Office связаны с соответствующими коллекциями. Обычно имя коллекции представляет собой множественное число от имени связанного объекта. Например, в коллекции CommandBars хранятся объекты CommandBar. Для простоты связанные коллекции и объекты будут описываться в одном разделе.

В большинстве случаев назначением коллекции является хранение одинаковых объектов. Общие свойства коллекций перечислены в следующем разделе. В соответствующих разделах будут рассматриваться только уникальные свойства, методы и события.

Общие свойства коллекций

Имя	*Тип возвращаемого значения*	*Описание*
Application	Application	Только чтение. Возвращает ссылку на приложение, которому принадлежит текущий объект
Count	Long	Только чтение. Возвращает количество объектов в коллекции

Имя	Тип возвращаемого значения	Описание
Creator	Long	Только чтение. Возвращает число типа Long, которое описывает приложение, создавшее объект. Это свойство используется только на платформе Macintosh
Parent	Object	Только чтение. Объект Parent является контейнером для объекта коллекции. Например, Workbooks.Parent возвращает ссылку на объект Application

Общие свойства объектов

Объекты также обладают общими свойствами. Ниже перечислены все общие свойства объектов. В разделах с описаниями объектов эти свойства будут упоминаться, но их определения приводятся только здесь.

Имя	Тип возвращаемого значения	Описание
Application	Application	Только чтение. Возвращает ссылку на приложение, которому принадлежит текущий объект
Creator	Long	Только чтение. Возвращает число типа Long, которое описывает приложение, создавшее объект. Это свойство используется только на платформе Macintosh
Parent	Object	Только чтение. Объект Parent является контейнером для текущего объекта. Например, в Excel вызов Shapes(1).Parent может вернуть ссылку на объект Worksheet, так как объект Worksheet является одним из возможных контейнеров объекта Shapes

Объекты пакета Office, их свойства и события

Объекты перечислены в алфавитном порядке. Для каждого объекта приводится общее описание и перечисляются возможные родительские объекты. После этого в табличном формате перечислены свойства и методы объекта. В последнем разделе описания приведены примеры использования объекта.

Объект AnswerWizard

Объект AnswerWizard является частью коллекции AnswerWizardFiles. Этот объект используется для управления файлами, на основе которых создается диалоговое окно Answer Wizard.

Общие свойства объекта AnswerWizard

Определения свойств Application, Creator и Parent приводились в начале приложения.

Свойства объекта AnswerWizard

Имя	Тип возвращаемого значения	Описание
Files	AnswerWizard-Files	Только чтение. Возвращает список файлов, доступных текущему объекту AnswerWizard

Методы объекта AnswerWizard

Имя	Тип возвращаемого значения	Параметры	Описание
ClearFileList			Очищает список файлов для текущего объекта AnswerWizard, включая список принятых по умолчанию файлов для текущего приложения Microsoft Office
ResetFileList			Сбрасывает список файлов для текущего объекта AnswerWizard и заменяет его принятым по умолчанию списком файлов для текущего приложения Microsoft Office

Коллекция AnswerWizardFiles

В этой коллекции хранятся ссылки на файлы мастера Answer Wizard. В коллекции находятся все файлы мастера Answer Wizard (это файлы с расширением .AW), доступные активному приложению Microsoft Office.

Общие свойства коллекции AnswerWizardFiles

Определения свойств Application, Count, Creator и Parent приводились в начале приложения.

Свойства коллекции AnswerWizardFiles

Имя	Тип возвращаемого значения	Параметры	Описание
Item	String	Index As Long	Только чтение. Возвращает строку с именем файла из коллекции AnswerWizardFiles

Методы коллекции AnswerWizardFiles

Имя	Тип возвращаемого значения	Параметры	Описание
Add		FileName as String	Создает новую ссылку (строковое значение) на файл мастера Answer Wizard и добавляет созданную ссылку в коллекцию AnswerWizardFiles
Delete		FileName as String	Удаляет указанный файл мастера Answer Wizard из коллекции AnswerWizardFiles

Объект Assistant

Этот объект управляет внешним видом и содержанием помощника Office. Например, объект Assistant можно использовать для вывода собственных сообщений вместо функции MsgBox. Многие свойства объекта Assistant соответствуют параметрам в диалоговом окне с параметрами помощника Office.

Общие свойства объекта Assistant

Определения свойств Application, Creator и Parent приводились в начале приложения.

Свойства объекта Assistant

Имя	*Тип возвращаемого значения*	*Описание*
Animation	MsoAnimation-Type	Устанавливает/возвращает анимацию для помощника Office
AssistWith-Alerts	Boolean	Устанавливает/возвращает необходимость вывода предупреждений приложения через всплывающие подсказки помощника, когда помощник Office виден на экране. При установке этого свойства в значение False предупреждения будут выводиться в принятых по умолчанию диалоговых окнах приложения
AssistWith-Help	Boolean	Устанавливает/возвращает необходимость отображения помощника Office при нажатии клавиши <F1> для доступа к справочному руководству. Если это свойство установлено в значение False, при нажатии клавиши <F1> выводится окно справочного руководства
BallonError	MsoBallon-ErrorType	Только чтение. Возвращает значение, соответствующее последней записанной ошибке из всплывающего сообщения
FeatureTips	Boolean	Устанавливает/возвращает необходимость предоставления информации о более эффективных методах использования приложения через помощника Office
FileName	String	Устанавливает/возвращает имя файла и расположение текущего помощника Office
GuessHelp	Boolean	Устанавливает/возвращает список тем справочного руководства, соответствующих ключевым словам, выделенным перед щелчком на окне помощника или нажатием клавиши <F1>. Если помощник Office отключен (см. описание свойства On), темы справочного руководства перечисляются в стандартном окне справочного руководства
High-PriorityTips	Boolean	Устанавливает/возвращает необходимость отображения высокоприоритетных подсказок через помощника Office
Item	String	Только чтение. Возвращает текст, связанный с объектом
Keyboard-ShortcutTips	Boolean	Устанавливает/возвращает необходимость вывода информации о комбинациях клавиш через помощника Office
Left	Long	Устанавливает/возвращает горизонтальное положение окна помощника Office (в пунктах) от левого края экрана
MouseTips	Boolean	Устанавливает/возвращает необходимость вывода информации об эффективном использовании мыши

Имя	*Тип возвращаемого значения*	*Описание*
MoveWhenIn-TheWay	Boolean	Устанавливает/возвращает необходимость автоматического перемещения окна помощника Office при попадании в рабочую область пользователя
Name	String	Только чтение. Возвращает имя текущего помощника Office. Если помощник отключен с помощью одного из параметров из диалогового окна Параметры помощника (Assistant Options) или с помощью свойства Assistant.On, свойство Name возвращает значение Nothing
NewBalloon	Balloon	Только чтение. Возвращает ссылку на объект Balloon. Хотя можно создавать несколько объектов Balloon, только один из них выводится на экран в определенный момент
On	Boolean	Устанавливает/возвращает состояние (включен/отключен) помощника Office
Reduced	Boolean	Устанавливает/возвращает необходимость отображения уменьшенного окна помощника Office. Это свойство не оказывает никакого действия на приложения Office XP
SearchWhen-Programming	Boolean	Устанавливает/возвращает необходимость вывода справочных материалов по приложениям и программированию, пока пользователь работает в редакторе VBE
Sounds	Boolean	Устанавливает/возвращает необходимость воспроизведения звуков вместе с соответствующей анимацией
TipOfDay	Boolean	Устанавливает/возвращает необходимость отображения специальной подсказки при каждом открытии приложения Office
Top	Long	Устанавливает/возвращает расстояние (в пунктах) от верхнего края помощника Office до верхнего края экрана
Visible	Boolean	Устанавливает/возвращает видимость помощника Office на экране. Это свойство не оказывает видимого эффекта, если свойство Assistant.On установлено в значение False

Методы объекта Assistant

Имя	*Тип возвращаемого значения*	*Параметры*	*Описание*
Activate-Wizard		WizardID as Long, Act as MsoWizard ActType, [Animation]	Восстанавливает или приостанавливает работу справочного руководства помощника Office во время работы собственного мастера. Этот метод используется только вместе с методом StartWizard

Имя	*Тип возвращаемого значения*	*Параметры*	*Описание*
DoAlert	Long	BstrAlertTitle as String, bstrAlertText as String, alb as MsoAlertButton Type, alc as MsoAlertIcon Type, ald as MsoAlertDefault Type, alq as MsoAlert CancelType, varfSysAlert as Boolean	Выводит предупреждение через помощника Office или через обычное окно сообщения. Возвращает значение типа Long, определяющее кнопку, на которой щелкнул пользователь. При включенном помощнике Office сообщение выводится в виде всплывающей подсказки. Если помощник Office отключен, сообщение выглядит и ведет себя как обычное окно сообщения. Этот метод очень похож на свойство Balloon, которое выводит сообщение только при включенном помощнике Office
EndWizard		WizardID as Long, varfSuc cess, Animation	Освобождает переменную, которая возвращается методом StartWizard. Этот метод используется только вместе с методом StartWizard
Help			Выводит помощника Office и шаблонное сообщение "What would you like to do?" для стандартного интерактивного справочного руководства Office
Move		XLeft as Integer, yTop as Integer	Перемещает помощника Office в указанную позицию. Помощник должен быть включен и видим на экране, но если одно из этих условий не выполняется, ошибки не возникает
ResetTips			Сбрасывает советы приложения, которые выводятся в окне помощника Office
StartWizard	Long	On as Boolean, Callback as String, PrivateX as Long, [Animation], [CustomTeaser], [Top], [Left], [Bottom], [Right]	Запускает помощника Office и возвращает значение типа Long, идентифицирующее сеанс. Этот метод необходимо использовать только для запуска помощника Office из собственного мастера. Возвращаемое этим методом число используется методами ActivateWizard и EndWizard

Пример: объект Assistant

В следующей подпрограмме некоторые свойства объекта `Assistant` используются для вывода помощника на экран, вывода вопроса к пользователю и обработки ответа пользователя.

```
' Открытый определенный пользователем тип, который используется для
' хранения текущих свойств помощника Office
Public Type AssistInfo
    bOn As Boolean
    bVisible As Boolean
    sFileName As String
End Type

Sub Opening()

    Dim lReturnValue As Long
    Dim oBalloon As Balloon
    Dim udAssistCurr As AssistInfo

    ' Сохранить текущие параметры помощника Office
    With udAssistCurr
        .bOn = Assistant.On
        .bVisible = Assistant.Visible
        .sFileName = Assistant.Filename
    End With

    ' Модифицировать помощника и вывести собственное всплывающее
    ' сообщение

    With Assistant
        .On = True
        .Visible = True
        .Filename = "C:\Program Files\Microsoft
↳ Office\Office10\dot.acs"
        .Animation = msoAnimationCheckingSomething

        ' Создать всплывающее сообщение
        Set oBalloon = .NewBalloon

        ' Модифицировать новое всплывающее сообщение
        With oBalloon
            .Heading = "Wrox Press Welcomes You"
            .Text = "Do you want to load the Wrox Press custom
↳ workbook?"
            .Button = msoButtonSetYesNo
            lReturnValue = .Show
        End With

        If lReturnValue = msoBalloonButtonYes Then ' Пользователь
        ' щелкнул на кнопке Да
            ' Открыть книгу и вывести собственную анимацию
            Workbooks.Open "C:\My Documents\Wrox\Wrox Examples.xls"
            .Animation = msoAnimationGetTechy
            .Animation = msoAnimationAppear
        Else
            ' Сбросить свойства помощника Office
            If udAssistCurr.bOn Then .Animation = msoAnimationGoodbye
            If Len(udAssistCurr.sFileName) Then _
                .Filename = udAssistCurr.sFileName
            .Visible = udAssistCurr.bVisible
```

```
            .On = udAssistCurr.bOn
        End If
    End With

End Sub
```

Объект Balloon

Этот объект используется для создания и вывода сообщений с помощью помощника Office. В большинстве случаев он применяется для создания собственного сообщения с текстом, метками, флажками и/или командными кнопками. Для этого объектной переменной присваивается значение свойства NewBalloon. Дополнительная информация приводится в описании объекта Assistant.

Общие свойства объекта Balloon

Определения свойств Application, Creator и Parent приводились в начале приложения.

Свойства объекта Balloon

Имя	*Тип возвращаемого значения*	*Описание*
Animation	MsoAnimationType	Устанавливает/возвращает операцию анимации помощника Office. При установке этого свойства объекта Balloon анимация помощника включается только при выводе помощника на экран
BalloonType	MsoBalloonType	Устанавливает/возвращает тип всплывающего сообщения, которое используется помощником Office. При создании объекта Balloon это свойство устанавливается в значение msoBalloonTypeButtons
Button	MsoButtonSetType	Устанавливает/возвращает тип и количество кнопок, которые выводятся в нижней части всплывающего сообщения помощника Office. При создании объекта Balloon это свойство устанавливается в значение msoButtonSetOK
Callback	String	Устанавливает/возвращает имя процедуры, которая вызывается из немодального всплывающего сообщения. При использовании свойства Callback с немодальными всплывающими сообщениями необходимо создать процедуру, принимающую три аргумента: объект Balloon, вызывающий процедуру, целое число типа Long, имеющее значение msoBalloonButtonType и описывающее кнопки, на которых щелкнул пользователь, а также целое число типа Long, идентифицирующее всплывающее сообщение, из которого была вызвана процедура (дополнительная информация приводится в описании свойства Private)
Heading	String	Устанавливает/возвращает текст заголовка, который выводится во всплывающем сообщении помощника Office
Icon	MsoIconType	Устанавливает/возвращает тип пиктограммы, которая выводится в верхнем левом углу всплывающего сообщения помощника Office

Имя	*Тип возвращаемого значения*	*Описание*
Labels	BalloonLabels	Только чтение. Возвращает коллекцию BalloonLabels, которая описывает метки кнопок, чисел и списков во всплывающем сообщении помощника Office
Mode	MsoModeType	Устанавливает/возвращает модальное поведение всплывающего сообщения помощника Office. При создании объекта Balloon это свойство устанавливается в значение msoModeModal, поэтому перед продолжением работы с приложением пользователь должен закрыть всплывающее сообщение
Name	String	Только чтение. Возвращает имя объекта Balloon
Private	Long	Устанавливает/возвращает целое число, идентифицирующее объект Balloon, вызвавший процедуру через свойство Callback. Используется для идентификации немодальных объектов Balloon при запуске процедуры через свойство Callback. Это свойство передается в процедуру в качестве одного из параметров
Text		Устанавливает/возвращает текст, который выводится напротив флажка или метки во всплывающем сообщении помощника Office. Свойство относится к объектам BalloonLabel и BalloonCheckBox

Методы объекта Balloon

Имя	*Тип возвращаемого значения*	*Параметры*	*Описание*
Close	Balloon		Закрывает активное немодальное всплывающее сообщение. Этот метод необходимо использовать только в процедурах, которые вызываются через свойство Callback
SetAvoid-Rectangle	Assistant	Left as long, Top as Long, Right as Long, Bottom as Long	Запрещает вывод всплывающего сообщения в пределах указанной области экрана
Show	MsoBalloon-ButtonType		Выводит указанный объект Balloon на экран

Коллекция BalloonCheckBoxes

В этой коллекции хранятся все флажки, которые выводятся в пределах окна объекта Balloon. Коллекция используется для перебора всех объектов BalloonCheckBox и определения состояния флажков на момент закрытия всплывающего сообщения.

Общие свойства коллекции BalloonCheckBoxes

Определения свойств Application, Count, Creator и Parent приводились в начале приложения.

Свойства коллекции BalloonCheckBoxes

Имя	*Тип возвращаемого значения*	*Описание*
`Item`	`BalloonCheckBox`	Только чтение. Возвращает элемент коллекции. Номер элемента коллекции указывается через параметр `Index`
`Name`	`String`	Только чтение. Возвращает имя объекта `BalloonCheckBox`

Объект BalloonCheckBox

Объект описывает флажки из коллекции флажков в пределах объекта `Balloon`.

Общие свойства объекта BalloonCheckBox

Определения свойства `Application`, `Creator` и `Parent` приводились в начале приложения.

Свойства объекта BalloonCheckBox

Имя	*Тип возвращаемого значения*	*Описание*
`Checked`	`Boolean`	Устанавливает/возвращает состояние конкретного флажка во всплывающем сообщении помощника Office
`Item`	`String`	Только чтение. Возвращает текст, связанный с объектом `BalloonCheckBox`
`Name`	`String`	Только чтение. Возвращает имя объекта `BalloonCheckBox`
`Text`		Устанавливает/возвращает текст, который выводится рядом с флажком во всплывающем сообщении помощника Office

Коллекция BalloonLabels

В этой коллекции хранятся метки, отображаемые в пределах объекта `Balloon`. Коллекция используется для перебора объектов `BalloonLabel` с целью определения установленных меток на момент закрытия сообщения.

Общие свойства коллекции BalloonLabels

Определения свойств `Application`, `Count`, `Creator` и `Parent` приводились в начале приложения.

Свойства коллекции BalloonLabels

Имя	*Тип возвращаемого значения*	*Описание*
`Item`	`BalloonCheckBox`	Только чтение. Возвращает элемент коллекции, номер которого указывается в качестве значения параметра `Index`
`Name`	`String`	Только чтение. Возвращает имя объекта `BalloonLabel`

Объект BalloonLabel

Представляет одну из меток, которые выводятся в пределах окна объекта `Balloon`.

Общие свойства объекта BalloonLabel

Определения свойств Application, Creator и Parent приводились в начале приложения.

Свойства объекта BalloonLabel

Имя	*Тип возвращаемого значения*	*Описание*
Checked	Boolean	Устанавливает/возвращает состояние флажка в пределах окна помощника Office
Item	String	Только чтение. Возвращает текст, связанный с объектом BalloonLabel
Name	String	Только чтение. Возвращает имя объекта BalloonLabel
Text	String	Устанавливает/возвращает текст, который выводится возле метки в пределах всплывающего сообщения помощника Office

Пример: коллекция BalloonCheckBoxes

В следующем примере коллекция BalloonCheckBoxes и объект BalloonCheckBox используются для вывода списка городов во всплывающем сообщении объекта Balloon. Результат выбора предоставляется с помощью метода DoAlert объекта Assistant.

```
Sub CustomBalloon()

   Dim lItem As Long
   Dim lReturnValue As Long
   Dim oBalloon As Balloon
   Dim oCheckBox As BalloonCheckbox
   Dim sMessage As String
   Dim vCities As Variant

   ' Создать массив городов, которые должны предоставляться во
   ' всплывающем сообщении
   vCities = Array("New York", "London", "Paris")

   ' Вывести собственное всплывающее сообщение
   With Assistant
      .On = True
      .Visible = True

      ' Создать новое всплывающее сообщение
      Set oBalloon = .NewBalloon

      ' Модифицировать новое всплывающее сообщение
      With oBalloon
         ' Добавить текст заголовка с выделением синим цветом и
         ' подчеркиванием
         .Heading = "{cf 252}{ul 1}Wrox Press{ul 0}{cf 0}"
         ' Добавить изображение
         ' и зеленый текст инструкции двумя строками ниже
         .Text = _
"{WMF ""C:\Program Files\Microsoft Office\Clipart\Office\TRAVEL.WMF""}" & _
                       vbCrLf & vbCrLf & _
                       "{cf 2}Выберите города, которые хотите
↳ посетить.{cf 0}"

            ' Создать набор флажков, текст которых соответствует
```

```
            ' названиям городов в массиве
            For lItem = 1 To 3
               ' По умолчанию нумерация массива начинается с 0
               .CheckBoxes(lItem).Text = vCities(lItem - 1)
            Next lItem

            ' Вывести всплывающее сообщение
            .Show

            ' Перебрать коллекцию BalloonCheckboxes и определить
            ' установленные флажки
            For Each oCheckBox In .CheckBoxes
               If oCheckBox.Checked Then _
                  sMessage = sMessage & oCheckBox.Text & vbCrLf
            Next oCheckBox

         End With

         ' Вывести результат с помощью метода .DoAlert
         If Len(sMessage) Then
            ' Пользователь выбрал как минимум один город
            sMessage = "Выбранные города:" & vbCrLf & vbCrLf & sMessage
         Else
            ' Пользователь не выбрал ни одного города
            sMessage = "Вы отшельник!"
         End If

      .DoAlert "{cf 2}Wrox{cf 0}", "{cf 1}" & sMessage & "{cf 0}", _
               msoAlertButtonOK, msoAlertIconInfo, _
               msoAlertDefaultFirst, msoAlertCancelDefault, False

   End With

End Sub
```

Коллекция COMAddins

В этой коллекции хранится список объектов COMAddin для приложений пакета Microsoft Office (в данном случае Excel). Надстройки COM предназначены для использования в различных приложениях Office, например Excel, Access, Word и Outlook. Надстройки COM разрабатываются на любом языке (Visual Basic, C++ или J++), поддерживающем создание компонентов COM (Component Object Model).

Общие свойства коллекции COMAddins

Определения свойств Application, Count, Creator и Parent приводились в начале приложения.

Методы коллекции COMAddins

Имя	*Тип возвращаемого значения*	*Параметры*	*Описание*
Item	COMAddIn	Index As Variant	Возвращает элемент коллекции COMAddins
Update			Обновляет содержимое коллекции COMAddins из списка надстроек, хранящегося в системном реестре Windows

Объект COMAddin

Объект соответствует одной надстройке приложения Microsoft Office и является членом коллекции COMAddins. Надстройки COM предназначены для использования в различных приложениях Office, например Excel, Access, Word и Outlook. Надстройки COM разрабатываются на любом языке (Visual Basic, C++ или J++), поддерживающем создание компонентов COM (Component Object Model).

Общие свойства объекта COMAddin

Определения свойств Application, Creator и Parent приводились в начале приложения.

Свойства объекта COMAddin

Имя	*Тип возвращаемого значения*	*Описание*
Connect	Boolean	Устанавливает/возвращает состояние подключения для объекта COMAddin
Description	String	Устанавливает/возвращает строку описания для объекта COMAddin
Guid	String	Только чтение. Возвращает глобально уникальный идентификатор класса (GUID) для объекта COMAddin
Object	Object	Устанавливает/возвращает объект, который лежит в основе объекта COMAddin. В основном это свойство используется для связи с другими объектами COMAddin
ProgId	String	Только чтение. Возвращает программный идентификатор (ProgID) объекта COMAddin

Пример: объект COMAddin

Следующая подпрограмма перебирает список надстроек COM и выводит информацию о надстройках в таблице на листе Sheet1 в книге, в которой находится код.

```
Sub COMAddinInfo()

    Dim lRow As Long
    Dim oCom As COMAddIn

    ' Установить заголовки на листе Sheet1 текущей книги
    With Sheet1.Range("A1:D1")
        .Value = Array("Guid", "ProgId", "Creator", "Description")
        .Font.Bold = True
        .HorizontalAlignment = xlCenter
    End With

    ' Перебрать коллекцию COMAddins и вывести информацию в ячейки
    ' под заголовками
    If Application.COMAddIns.Count Then
        For Each oCom In Application.COMAddIns

            With Sheet1.Range("A2")
                .Offset(lRow, 0).Value = oCom.GUID
                .Offset(lRow, 1).Value = oCom.progID
                .Offset(lRow, 2).Value = oCom.Creator
                .Offset(lRow, 3).Value = oCom.Description
                lRow = lRow + 1
```

```
            End With
        Next oCom
    End If

    ' Автоматически изменить ширину столбцов таблицы
    Sheet1.Range("A1:D1").EntireColumn.AutoFit

End Sub
```

Коллекция CommandBars

В коллекции CommandBars хранится список всех объектов CommandBar (большинство пользователей знают эти объекты как панели инструментов) в пределах приложения. Дополнительная информация приводится в описании объекта CommandBar. Коллекция обеспечивает доступ к параметрам, предоставленным на вкладке Параметры (Options) в диалоговом окне Настройка (Customize).

Общие свойства коллекции CommandBars

Определения свойств Application, Count, Creator и Parent приводились в начале приложения.

Свойства коллекции CommandBars

Имя	*Тип возвращаемого значения*	*Описание*
ActionnewlineControl	CommandBarControl	Только чтение. Возвращает объект CommandBarControl, свойство OnAction которого указывает на работающую процедуру. Если работающая процедура не была запущена элементом управления командной панели, свойство возвращает значение Nothing
ActiveMenuBar	CommandBar	Только чтение. Возвращает объект CommandBar, который соответствует активной панели меню в приложении. Практически всегда это свойство возвращает меню листа
AdaptiveMenus	Boolean	Устанавливает/возвращает необходимость использования адаптивных меню
DisableAskAQuestionDxropdown	Boolean	Устанавливает/возвращает состояние раскрывающегося меню мастера Answer Wizard, которое выводится справа от меню листа в приложении. При установке этого свойства в значение True раскрывающийся список исчезает с панели меню
DisableCustomize	Boolean	Устанавливает/возвращает необходимость отключения панели инструментов с параметрами настройки. Установка этого свойства в значение True приводит к отключению пункта Настройка (Customize) в меню Сервис (Tools) и в контекстом меню панелей управления
DisplayFonts	Boolean	Устанавливает/возвращает необходимость использования реальных шрифтов при заполнении раскрывающегося списка Шрифт (Font). На старых компьютерах с ограниченными вычислительными ресурсами это свойство стоит установить в значение False

Имя	Тип возвращаемого значения	Описание
DisplayKeysIn-Tooltips	Boolean	Устанавливает/возвращает необходимость вывода комбинаций клавиш в подсказках для элементов управления командных панелей. Это свойство не оказывает влияния на командные панели Excel
Display-Tooltips	Boolean	Устанавливает/возвращает необходимость вывода всплывающих подсказок при наведении указателя мыши на элементы управления командных панелей
Item	CommandBar	Только чтение. Возвращает объект CommandBar из коллекции CommandBars. Номер объекта указывается в качестве значения параметра Index. Кроме этого, в качестве значения параметра Index можно указывать имя командной панели
LargeButtons	Boolean	Устанавливает/возвращает необходимость вывода больших кнопок на панелях инструментов
MenuAnimation-Style	Mso-MenuAnimation	Устанавливает/возвращает тип анимации для всех элементов управления CommandBarPopup (меню) в коллекции CommandBars

Методы коллекции CommandBars

Имя	Тип возвращаемого значения	Параметры	Описание
Add	CommandBar	[Name], [Position], [MenuBar], [Temporary]	Создает новую командную панель и добавляет ее в коллекцию CommandBars
FindControl	CommandBar-Control	[Type], [Id], [Tag], [Visible]	Возвращает один объект CommandBarControl, соответствующий описанному параметрами критерию
FindControls	CommandBar-Controls	[Type], [Id], [Tag], [Visible]	Возвращает из коллекции набор объектов CommandBarControl, соответствующих указанному с помощью параметров критерию
ReleaseFocus			Снимает фокус пользовательского интерфейса со всех командных панелей

События коллекции CommandBars

Имя	Параметры	Описание
OnUpdate		Событие OnUpdate распознается объектом CommandBar и всеми элементами управления командных панелей. Из-за большого количества событий OnUpdate, возникающих в процессе нормальной работы, разработчики приложений для Excel должны использовать это событие с осторожностью

Пример: коллекция CommandBars

Следующая подпрограмма устанавливает параметры всех объектов CommandBar. После этого выводится количество командных панелей в приложении, а также имя активной панели меню.

```
Sub CountCommandBars()

    ' Модифицировать некоторые параметры всех командных панелей
    With CommandBars
        ' Включить адаптивные меню
        .AdaptiveMenus = True

        ' Удалить поле Справка, которое выводится в правой части меню
        ' Обратите внимание: Этот параметр не затрагивает поле
        ' Справка в редакторе VBE
        .DisableAskAQuestionDropdown = True

        ' Запретить модификацию командных панелей
        .DisableCustomize = True

        ' Не показывать внешний вид шрифтов в раскрывающемся списке Шрифт
        ' Это позволяет сохранить ресурсы и повысить быстродействие
        .DisplayFonts = False

        ' Включить случайную анимацию при щелчке на пункте меню
        .MenuAnimationStyle = msoMenuAnimationRandom

        ' Выводить подсказку при наведении указателя мыши
        .DisplayTooltips = True

        ' Показывать комбинации клавиш вместе с подсказками (не
        ' оказывает влияния на Excel)
        .DisplayKeysInTooltips = True

        ' Выводить большие кнопки на командных панелях
        .LargeButtons = True

        ' Вывести количество командных панелей и название активного
        ' меню.
        MsgBox "В приложении " & .Parent.Name & " доступно " & .Count
➥ & " командных панелей." _
                & "В данный момент активно меню " _

                & .ActiveMenuBar.Name & "." _
                , vbInformation, "Wrox"

    End With

End Sub
```

Объект CommandBar

В этом объекте хранятся свойства и методы конкретной командной панели из коллекции CommandBars. Свойства и методы этого объекта похожи на свойства и методы коллекции CommandBars, но они относятся только к конкретной командной панели. Для получения объекта CommanBar воспользуйтесь вызовом CommandBars(Index), например:

```
Dim oBar As CommandBar
Set oBar = CommandBars("Wrox")
```

Общие свойства объекта CommandBar

Определения свойств Application, Creator и Parent приводились в начале приложения.

Свойства объекта CommandBar

Имя	*Тип возвращаемого значения*	*Описание*
AdaptiveMenu	Boolean	Только чтение. Возвращает объект CommandBar, который соответствует активной панели меню в приложении
BuiltIn	Boolean	Только чтение. Возвращает значение True, если указанная панель меню или элемент панели меню являются встроенными. Возвращает значение False, если это собственная командная панель или собственный элемент командной панели. Кроме этого, значение False возвращается для встроенных элементов управления с модифицированным значением свойства OnAction
Context	String	Устанавливает/возвращает строку, которая определяет место сохранения командной панели. Строка определяется и интерпретируется приложением
Controls	CommandBar-Controls	Только чтение. Возвращает объект CommandBarControl, представляющий все элементы управления на командной панели
Enabled	Boolean	Устанавливает/возвращает состояние командной панели (включена/выключена). Установка этого свойства в значение True приводит к появлению имени командной панели в списке доступных командных панелей
Height	Long	Устанавливает/возвращает высоту командной панели
ID	Long	Только чтение. Возвращает идентификатор встроенной командной панели
Index	Long	Только чтение. Возвращает положение объекта CommandBar в коллекции CommandBars
Left	Long	Устанавливает/возвращает расстояние (в пикселях) от левого края командной панели до левого края экрана
Name	String	Устанавливает/возвращает имя объекта CommandBar
NameLocal	String	Устанавливает/возвращает имя встроенной командной панели, которое выводится в локализованной языковой версии приложения, а также имя собственной командной панели
Position	MsoBarPosition	Устанавливает/возвращает положение командной панели
Protection	MsoBar-Protection	Устанавливает/возвращает способ защиты командной панели от пользовательской модификации
RowIndex	Long	Устанавливает/возвращает порядок размещения командной панели относительно других командных панелей в пределах одной области. Свойству можно присвоить целое значение больше нуля или одну из констант MsoBarRow: msoBarRowFirst или msoBarRowLast
Top	Long	Устанавливает/возвращает расстояние (в пунктах) от верхнего края командной панели до верхнего края экрана

Имя	*Тип возвращаемого значения*	*Описание*
Type	MsoBarType	Только чтение. Возвращает тип командной панели
Visible	Boolean	Устанавливает/возвращает видимость командной панели. Перед установкой этого свойства в значение True в это же значение должно быть установлено свойство Enabled
Width	Long	Устанавливает/возвращает ширину (в пикселях) командной панели

Методы объекта CommandBar

Имя	*Тип возвращаемого значения*	*Параметры*	*Описание*
Delete			Удаляет объект CommandBar из коллекции CommandBars
FindControl	CommandBar-Control	[Type], [Id], [Tag], [Visible], [Recursive]	Возвращает объект CommandBar, соответствующий указанному критерию
Reset			Сбрасывает объект CommandBar в принятую по умолчанию конфигурацию
ShowPopup		[x], [y]	Выводит объект CommandBar в виде контекстного меню по указанным координатам или по текущим координатам указателя

Пример: объект CommandBar

Следующая подпрограмма модифицирует собственную командную панель (объект CommandBar), которая называется "Wrox".

```
Sub CustomizeWroxBar()

    Dim oBar As CommandBar
    Dim oMenu As CommandBarPopup

    ' Убедится в существовании командной панели
    On Error Resume Next
    Set oBar = Application.CommandBars("Wrox")

    On Error GoTo Error

    ' Если командная панель существует, выполнить модификацию
    If Not oBar Is Nothing Then
        With oBar
            ' Добавить разделитель перед вторым элементом управления
            .Controls(2).BeginGroup = True

            ' Сместить панель вправо
            .Position = msoBarRight

            ' Запретить модификацию пользователем
            .Protection = msoBarNoCustomize

            ' Сделать панель видимой
```

```
            .Visible = True

            ' Получить ссылку на первый элемент управления
            ' командной панели Wrox.
            ' Это будет меню с дополнительными элементами управления
            Set oMenu = CommandBars("Wrox").Controls(1)
            With oMenu
                ' Изменить текст меню
                .Caption = "Member Info"

                ' Изменить текст всплывающей подсказки
                .TooltipText = "Insert Member Info"

                ' Добавить разделитель перед третим элементом
                ' управления в меню
                .Controls(3).BeginGroup = True
            End With

        End With
    Else
        ' Командная панель "Wrox" не существует.
        ' Вывести сообщение об ошибке
        Err.Raise Number:=glERROR_CUSTOM, _
                  Description:="Командная панель Wrox не найдена."
    End If

    Exit Sub

Error:
    ' Вывести сообщение об ошибке
    MsgBox Err.Number & vbLf & Err.Description

End Sub
```

Объект CommandBarButton

Объект CommandBarButton соответствует любой кнопке или пункту меню на каждой командной панели. Для получения доступа к конкретному объекту CommandBarButton необходимо идентифицировать объект CommandBar, в пределах которого находится кнопка, и воспользоваться свойством Controls(Index). Параметру Index можно присвоить порядковый номер объекта CommandBarButton в меню или на панели инструментов, или значение свойства Caption.

Например, к первому элементу управления, который называется "Wrox", можно обратиться с помощью оператора:

```
CommandBars("Wrox").Controls(1)
```

или

```
CommandBars("Wrox").Controls("Member Info")
```

Общие свойства объекта CommandBarButton

Определения свойств Application, Creator и Parent приводились в начале приложения.

Свойства объекта CommandBarButton

Имя	*Тип возвращаемого значения*	*Описание*
BeginGroup	Boolean	Устанавливает/возвращает информацию о том, расположен ли объект CommandBarButton в начале группы элементов управления на командной панели
BuiltIn	Boolean	Только чтение. Возвращает значение True, если указанная командная панель или элементы управления командной панели являются встроенными. Возвращает значение False, если это собственная командная панель или элемент управления. Кроме этого, значение False возвращается для встроенных элементов управления с модифицированным значением свойства OnAction
BuiltInFace	Boolean	Устанавливает/возвращает информацию об использовании встроенной пиктограммы элемента управления CommandBarButton. Это свойство устанавливается только в значение True, что приводит к сбросу пиктограммы и использованию встроенной пиктограммы элемента управления
Caption	String	Устанавливает/возвращает текст заголовка элемента управления CommandBarButton
DescriptionText	String	Устанавливает/возвращает описание объекта CommandBarButton. Описание не предоставляется пользователю, но может применяться для документирования поведения элемента управления для других разработчиков
Enabled	Boolean	Устанавливает/возвращает состояние элемента управления CommandBarButton (включен/выключен)
FaceId	Long	Устанавливает/возвращает номер идентификатора пиктограммы, которая используется объектом CommandBarButton
Height	Long	Устанавливает/возвращает высоту объекта CommandBarButton
HelpContextId	Long	Устанавливает/возвращает идентификатор контекста в справочном руководстве для объекта CommandBarButton
HelpFile	String	Устанавливает/возвращает имя файла справочного руководства для объекта CommandBarButton
HyperlinkType	MsoCommandBarButtonHyperlinkType	Устанавливает/возвращает тип гиперссылки, связанной с объектом CommandBarButton
Id	Long	Только чтение. Возвращает идентификатор встроенного объекта CommandBarButton
Index	Long	Только чтение. Возвращает положение объекта CommandBarButton в коллекции CommandBars
IsPriorityDropped	Boolean	Только чтение. Сообщает, что объект CommandBarButton исключен из меню или панели инструментов на основе статистики использования и площади макета. (Обратите внимание, что это свойство не идентично видимости элемента управления, которая устанавливается с помощью свойства Visible.) Если свойство IsPriorityDropped установлено в значение True, то установка свойства Visible в это же значение не приведет к немедленному появлению элемента управления на экране

Имя	*Тип возвращаемого значения*	*Описание*
Left	Long	Только чтение. Возвращает горизонтальное положение (в пикселях) объекта CommandBarButton относительно левого края экрана. Расстояние отсчитывается от левого края области присоединения
Mask	IPictureDisp	Возвращает объект IPictureDisp, соответствующий изображению маски объекта CommandBarButton. Изображение маски определяет прозрачные фрагменты пиктограммы кнопки
OLEUsage	MsoControl-OLEUsage	Устанавливает/возвращает роли клиента OLE и сервера OLE, в которых будет использоваться объект CommandBarButton при объединении двух приложений Microsoft Office
OnAction	String	Устанавливает/возвращает имя процедуры Visual Basic, которая запускается при щелчке пользователя или при изменении значения объекта CommandBarButton
Parameter	String	Устанавливает/возвращает строку, которая может использоваться приложением для выполнения команды
Picture	IPictureDisp	Устанавливает/возвращает объект IPictureDisp, который описывает пиктограмму объекта CommandBarButton
Priority	Long	Устанавливает/возвращает приоритет объекта CommandBarButton
Shortcut-Text	String	Устанавливает/возвращает текст комбинации клавиш, который выводится напротив элемента управления CommandBarButton в меню, подменю или в контекстном меню
State	MsoButton-State	Устанавливает/возвращает внешний вид объекта CommandBarButton
Style	MsoButton-Style	Устанавливает/возвращает способ вывода объекта CommandBarButton на экран
Tag	String	Устанавливает/возвращает информацию об объекте CommandBarButton. Например, значение этого свойства может использоваться в качестве аргумента при вызове процедур
TooltipText	String	Устанавливает/возвращает текст, который выводится в виде всплывающей подсказки для объекта CommandBarButton
Visible	Boolean	Устанавливает/возвращает видимость объекта CommandBarButton на экране
Width	Long	Устанавливает/возвращает ширину (в пикселях) указанного объекта CommandBarButton

Методы объекта CommandBarButton

Имя	*Тип возвращаемого значения*	*Параметры*	*Описание*
Copy	CommandBar-Button	[Bar], [Before]	Копирует объект CommandBarButton на существующую командную панель
CopyFace			Копирует пиктограмму объекта CommandBarButton в буфер обмена

Имя	*Тип возвращаемого значения*	*Параметры*	*Описание*
Delete		[Temporary]	Удаляет объект CommandBarButton из коллекции. Параметр Temporary устанавливается в значение True для удаления элемента управления только в пределах текущего сеанса. В следующем сеансе элемент управления опять будет показан на экране
Execute			Запускает процедуру или встроенную команду, связанную с объектом CommandBarButton. Для собственных элементов управления при выборе запускаемой процедуры можно воспользоваться свойством OnAction
Move	CommanBar-Control	[Bar], [Before]	Перемещает объект CommandBarButton на указанную командную панель
PasteFace			Вставляет содержимое буфера обмена в объект CommandBarButton
Reset			Сбрасывает встроенный объект CommandBarButton в принятую по умолчанию конфигурацию или устанавливает оригинальную функцию и пиктограмму объекта CommandBarButton
SetFocus			Перемещает фокус клавиатуры на указанный объект CommandBarButton. Если элемент управления отключен или не видим на экране, вызов метода завершится ошибкой

События объекта CommandBarButton

Имя	*Параметры*	*Описание*
Click	ByVal Ctrl As CommandBarButton, ByVal CancelDefault As Boolean	Возникает при щелчке пользователя на объекте CommandBarButton. Параметр Ctrl указывает на элемент управления, который сгенерировал событие. Если выполняется принятая по умолчанию связанная с элементом управления операция, параметр CancelDefault устанавливается в значение False. Значение этого параметра может измениться в результате работы другого процесса или надстройки

Пример: объект CommandBarButton

Первая подпрограмма создает объект CommandBarButton на собственной командной панели Wrox. Элементу управления назначается текст всплывающей подсказки и пиктограмма кнопки. После этого кнопка связывается с подпрограммой SelectNumericValues.

```
Sub AddCommandBarButton()

    Const sSELECT_NUMERIC As String = "SelectNumericValues"

    Dim ctlButton As CommandBarButton
```

```
    ' Добавить объект CommandBarButton
    Set ctlButton = CommandBars("Wrox").Controls.Add(msoControlButton)

    With ctlButton

        ' Назначить пиктограмму как у кнопки Select Visible Cells
        ' button в Excel
        .FaceId = 441

        ' Добавить текст всплывающей подсказки
        .TooltipText = "Select Numeric Values"

        ' Сохранить имя в свойстве Tag для обнаружения элемента
        ' управления методом FindControl без информации о его положении
        .Tag = sSELECT_NUMERIC
        ' Назначить подпрограмму элементу управления CommandBarButton
        .OnAction = sSELECT_NUMERIC
    End With

End Sub

Sub SelectNumericValues()

    Dim rng As Range

    ' Подавить сообщения об ошибках (например, отсутствие активной
    ' книги/листа)
    On Error Resume Next

    ' Убедится, что выделен диапазон
    If TypeOf Selection Is Range Then

        If Selection.Cells.Count = 1 Then
            ' Если выделена только одна ячейка,
            ' получить все числовые константы листа
            ActiveSheet.UsedRange.SpecialCells( _
                xlCellTypeConstants, xlNumbers).Select
        Else
            ' Получить числовые константы выделения
            Selection.SpecialCells(xlCellTypeConstants, xlNumbers).Select
        End If
    End If

End Sub
```

Объект CommandBarComboBox

Данный объект соответствует раскрывающемуся списку, текстовому полю или комбинированному списку (комбинация первых двух вариантов). Эти элементы управления могут существовать только на командной панели. При этом командная панель присоединяется к другим командным панелям либо находится в произвольном положении вверху или внизу окна приложения.

Общие свойства объекта CommandBarComboBox

Определения свойств Application, Creator и Parent приводились в начале приложения.

Свойства объекта CommandBarComboBox

Имя	*Тип возвращаемого значения*	*Описание*
BeginGroup	Boolean	Устанавливает/возвращает информацию о том, расположен ли объект CommandBarComboBox в начале группы элементов управления на командной панели
BuiltIn	Boolean	Только чтение. Возвращает значение True, если указанная командная панель или элементы управления командной панели являются встроенными. Возвращает значение False, если это собственная командная панель или элемент управления. Кроме этого, значение False возвращается для встроенных элементов управления с модифицированным значением свойства OnAction
Caption	String	Устанавливает/возвращает текст заголовка элемента управления CommandBarComboBox
Description-Text	String	Устанавливает/возвращает описание объекта CommandBarComboBox. Описание не предоставляется пользователю, но может применяться для документирования поведения элемента управления для других разработчиков
DropDown-Lines	Long	Устанавливает/возвращает количество строк в элементе управления CommandBarComboBox. Обратите внимание, что при попытке установить значение этого свойства для элемента управления с полем ввода или для встроенного элемента управления комбинированного списка возникает сообщение об ошибке
DropDown-Width	Long	Устанавливает/возвращает ширину (в пикселях) списка в элементе управления CommandBarComboBox. Обратите внимание, что при попытке установки этого свойства для встроенного элемента управления возникает сообщение об ошибке
Enabled	Boolean	Устанавливает/возвращает состояние элемента управления CommandBarComboBox (включен/выключен)
Height	Long	Устанавливает/возвращает высоту объекта CommandBarComboBox
Help-ContextId	Long	Устанавливает/возвращает идентификатор контекста для раздела справочного руководства, связанного с элементом управления CommandBarComboBox
HelpFile	String	Устанавливает/возвращает имя файла справочного руководства, связанного с элементом управления CommandBarComboBox
Index	Long	Только чтение. Возвращает значение типа Long, описывающее положение объекта CommandBarComboBox в коллекции CommandBars
IsPriority-Dropped	Boolean	Только чтение. Сообщает, что объект CommandBarComboBox исключен из меню или панели инструментов на основе статистики использования и площади макета. (Обратите внимание, что это свойство не идентично видимости элемента управления, которая устанавливается с помощью свойства Visible.) Если свойство IsPriorityDropped установлено в значение True, установка свойства Visible в это же значение не приведет к немедленному появлению элемента управления на экране

Имя	*Тип возвращаемого значения*	*Описание*
Left	Long	Только чтение. Возвращает горизонтальное положение (в пикселях) элемента управления CommandBarComboBox относительно левого края экрана. Возвращает расстояние от левого края области присоединения
List	String	Устанавливает/возвращает указанный элемент из списка CommandBarComboBox. Для встроенных элементов управления CommandBarComboBox возможно только чтение. Требует указания параметра: Index as Long
ListCount	Long	Только чтение. Возвращает количество элементов в списке объекта CommandBarComboBox
ListHeader-Count	Long	Устанавливает/возвращает количество элементов в списке объекта CommandBarComboBox, выводимых до разделителя. Свойство предназначено для использования со встроенными элементами управления ComboBox
ListIndex	Long	Устанавливает/возвращает номер выделенного элемента списка из объекта CommandBarComboBox. Если не выделен ни один из элементов, свойство возвращает значение 0
OLEUsage	MsoControl-OLEUsage	Устанавливает/возвращает роли OLE-клиента и OLE-сервера, в которых будет использоваться объект CommandBarComboBox при объединении двух приложений Microsoft Office
OnAction	String	Устанавливает/возвращает имя процедуры Visual Basic, которая запускается при щелчке пользователя или изменении значения объекта CommandBarComboBox
Parameter	String	Устанавливает/возвращает строку, которая используется приложением при вызове команд
Priority	Long	Устанавливает/возвращает приоритет объекта CommandBarComboBox
Style	MsoComboStyle	Устанавливает/возвращает способ вывода элемента управления CommandBarComboBox на экран. Для установки свойства используются константы MsoComboStyle: msoComboLabel и msoComboNormal
Tag	String	Устанавливает/возвращает информацию об элементе управления CommandBarComboBox, например, свойство может содержать данные, которые передаются в процедуры в качестве значений аргументов
Text	String	Устанавливает/возвращает текст в поле выбора или поле ввода элемента управления CommandBarComboBox
TooltipText	String	Устанавливает/возвращает текст всплывающей подсказки для элемента управления CommandBarComboBox
Top	Long	Только чтение. Возвращает расстояние (в пикселях) от верхнего края элемента управления CommandBarComboBox до верхнего края экрана
Type	MsoControl-Type	Только чтение. Возвращает тип элемента управления CommandBarComboBox
Visible	Boolean	Устанавливает/возвращает видимость элемента управления CommandBarComboBox
Width	Long	Устанавливает/возвращает ширину (в пикселях) элемента управления CommandBarComboBox

Методы объекта CommandBarComboBox

Имя	*Тип возвращаемого значения*	*Параметры*	*Описание*
AddItem		Text as String, [Index as Variant]	Добавляет элемент списка в указанный объект CommandBarComboBox. Это должен быть собственный раскрывающийся список или комбинированный список. При вызове этого метода для поля ввода или встроенного элемента управления работа метода завершится ошибкой
Clear			Удаляет все элементы из объекта CommandBarComboBox (раскрывающегося списка или комбинированного списка) или очищает содержимое текстового поля (поля ввода или комбинированного списка). При вызове этого метода для встроенного элемента управления работа метода завершится ошибкой
Copy	Object	[Bar], [Before]	Копирует элемент управления CommandBarComboBox в существующую командную панель
Delete		[Temporary]	Удаляет указанный объект CommandBarComboBox из коллекции. Параметр Temporary устанавливается в значение True для удаления элемента управления только на время текущего сеанса. После начала следующего сеанса элемент управления опять будет присутствовать на командной панели
Execute			Вызывает процедуру или встроенную команду, назначенную указанному элементу управления CommandBarComboBox. Для собственных элементов управления выбор процедуры осуществляется с помощью свойства OnAction
Move	CommandBar-Control	[Bar], [Before]	Перемещает указанный элемент управления CommandBarComboBox на существующую командную панель
RemoveItem		Index As Long	Удаляет указанный элемент из элемента управления CommandBarComboBox
Reset			Сбрасывает встроенный элемент управления CommandBarComboBox в принятую по умолчанию конфигурацию, восстанавливая функцию и пиктограмму
SetFocus			Перемещает фокус клавиатуры на указанный элемент управления CommandBar-ComboBox. Если элемент управления отключен или не показан на экране, вызов метода завершится ошибкой

События объекта CommandBarComboBox

Имя	*Параметры*	*Описание*
Change	ByVal Ctrl As CommandBarComboBox	Возникает при изменении выделения в элементе управления CommandBarComboBox

Пример: объект CommandBarComboBox

Следующая подпрограмма добавляет элемент управления CommandBarComboBox в собственную командную панель, которая называется Wrox. Список элемента управления заполняется адресами электронной почты, которые хранятся на листе книги, содержащей код. Подпрограмма MailTo связывается с комбинированным списком и используется для отправки активной книги выбранному из списка адресату.

```
Sub AddComboBox()
        Const sTAG_RUN As String = "MailTo"

    Dim ctlCombo As CommandBarComboBox
    Dim lItem As Long
    Dim vaItems As Variant
    Dim szttt As String

    ' Получить список элементов списка на листе книги
    vaItems = wksListData.Range("Items")

    ' Добавить элемент управления CommandBarComboBox
    Set ctlCombo = CommandBars("Wrox").Controls.Add(msoControlComboBox)

    With ctlCombo
        ' Добавить список элементов с листа в элемент управления
        ' CommandBarComboBox
        For lItem = LBound(vaItems) To UBound(vaItems)
            .AddItem vaItems(lItem, 1), lItem
        Next lItem

        ' Добавить текст всплывающей подсказки
        .TooltipText = "Send Workbook To"

        ' Сохранить имя в свойстве Tag, которое может использоваться
        ' методом FindControl для обнаружения элемента управления без
        ' информации о его позиции
        .Tag = sTAG_RUN

        ' Связать подпрограмму с элементом управления
        ' CommandBarComboBox
        .OnAction = sTAG_RUN
    End With

End Sub

Sub MailTo()

    Dim ctlCombo As CommandBarComboBox
    Dim lChoice As Long

    ' Подавить вывод сообщений об ошибках при отсутствии активной
    ' книги
    On Error Resume Next

    ' Получить доступ к элементу управления
```

```
    Set ctlCombo = CommandBars.ActionControl

    ' Какой пункт списка выбрал пользователь?
    lChoice = ctlCombo.ListIndex

    If lChoice Then
        ' Пользователь выбрал адресата. Отправить выбранному
        ' адресату активную книгу
        ActiveWorkbook.SendMail _
            Recipients:=ctlCombo.List(lChoice), _
            Subject:=ActiveWorkbook.Name, ReturnReceipt:=True
    End If

End Sub
```

Коллекция CommandBarControls

В этой коллекции хранятся все элементы управления командной панели. Имя коллекции используется только при определении типа переменной. Для непосредственного доступа ко всем элементам управления командной панели можно воспользоваться строкой

```
CommandBars(shape Index).Controls
```

где `Index` является числом, описывающим положение командной панели в коллекции, или строкой, содержащей имя командной панели.

Общие свойства коллекции CommandBarControls

Определения свойств `Application`, `Count`, `Creator` и `Parent` приводились в начале приложения.

Свойства коллекции CommandBarControls

Имя	*Тип возвращаемого значения*	*Параметры*	*Описание*
`Item`	`Object`	`Index As Variant`	Возвращает объект `CommandBarControl` из коллекции `CommandBarControls`

Методы коллекции CommandBarControls

Имя	*Тип возвращаемого значения*	*Параметры*	*Описание*
`Add`	`CommandBarControl`	`[Type], [Id], [Parameter], [Before], [Temporary]`	Создает новый объект `CommandBarControl` и добавляет его в коллекцию элементов управления указанной командной панели

Пример: коллекция CommandBarControls

Следующая подпрограмма перечисляет все элементы управления на собственной командной панели, которая называется Wrox. Кроме списка, на листе выводится информация о свойствах.

```
Sub ListAllControls()

    Dim ctl As CommandBarControl
    Dim ctlAll As CommandBarControls
```

```
    Dim lRow As Long

    ' Сохранить все элементы управления командной панели Wrox
    Set ctlAll = CommandBars("Wrox").Controls

    ' Инициализировать счетчик строк
    lRow = 2

    ' На листе в этой книге...
    With wksControls
        ' Удалить старый список
        .UsedRange.ClearContents

        ' Разместить заголовки на листе
        .Cells(1, 1).Value = "CAPTION"
        .Cells(1, 2).Value = "BUILTIN"
        .Cells(1, 3).Value = "ID"
        .Cells(1, 4).Value = "TAG"
        .Cells(1, 5).Value = "TOOLTIP"
        .Cells(1, 6).Value = "TYPE"

        ' Перебрать все элементы управления, выводя информацию о
        ' каждом элементе управления в соответствующих столбцах листа
        For Each ctl In ctlAll
            .Cells(lRow, 1).Value = ctl.Caption
            .Cells(lRow, 2).Value = ctl.BuiltIn
            .Cells(lRow, 3).Value = ctl.ID
            .Cells(lRow, 4).Value = ctl.Tag
            .Cells(lRow, 5).Value = ctl.TooltipText
            .Cells(lRow, 6).Value = ctl.Type
            lRow = lRow + 1 ' Увеличить значение счетчика строк
        Next

        ' Автоматически установить ширину столбцов
        .UsedRange.EntireColumn.AutoFit
    End With
End Sub
```

Объект CommandBarControl

Этот объект представляет универсальный элемент управления на командной панели. Обычно в качестве элемента управления выступает CommandBarButton, CommandBar-ComboBox или CommandBarPopup. При использовании одного из этих элементов управления с ним можно работать через отдельную объектную ссылку. При этом будут доступны все свойства и методы конкретного элемента управления.

Объект CommandBarControl применяется тогда, когда нет информации о типе используемого элемента управления или когда используется элемент управления, отличный от перечисленных выше. Большинство методов и свойств объекта CommandBarControl доступны для объектов CommandBarButton, CommandBarButton и CommandBarComboBox.

Общие свойства объекта CommandBarControl

Определения свойств Application, Creator и Parent приводились в начале приложения.

Свойства объекта CommandBarControl

Имя	*Тип возвращаемого значения*	*Описание*
BeginGroup	Boolean	Устанавливает/возвращает информацию о том, расположен ли объект CommandBarControl в начале группы элементов управления на командной панели
BuiltIn	Boolean	Только чтение. Возвращает значение True, если указанная командная панель или элементы управления командной панели являются встроенными. Возвращает значение False, если это собственная командная панель или элемент управления. Кроме этого, значение False возвращается для встроенных элементов управления с модифицированным значением свойства OnAction
Caption	String	Устанавливает/возвращает текст заголовка элемента управления CommandBarControl
Description-Text	String	Устанавливает/возвращает описание объекта CommandBarControl. Описание не предоставляется пользователю, но может применяться для документирования поведения элемента управления для других разработчиков
Enabled	Boolean	Устанавливает/возвращает состояние объекта CommandBarControl (включен/выключен)
Height	Long	Устанавливает/возвращает высоту элемента управления CommandBarControl
Help-ContextId	Long	Устанавливает/возвращает идентификатор контекста для раздела справочного руководства, связанного с элементом управления CommandBarControl
HelpFile	String	Устанавливает/возвращает имя файла справочного руководства, связанного с элементом управления CommandBarControl
Id	Long	Только чтение. Возвращает идентификатор встроенного элемента управления CommandBarControl
Index	Long	Только чтение. Возвращает число типа Long, являющееся номером объекта CommandBarControl в коллекции CommandBarControls
IsPriority-Dropped	Boolean	Только чтение. Сообщает, что объект CommandBarControl исключен из меню или панели инструментов на основе статистики использования и площади макета. (Обратите внимание, что это свойство не идентично видимости элемента управления, которая устанавливается с помощью свойства Visible.) Если свойство IsPriorityDropped установлено в значение True, установка свойства Visible в значение True не приведет к немедленному появлению элемента управления на экране
Left	Long	Только чтение. Возвращает горизонтальное положение (в пикселях) элемента управления CommandBarControl относительно левого края экрана. Возвращается расстояние от левого края области присоединения

Имя	*Тип возвращаемого значения*	*Описание*
`OLEUsage`	`MsoControl-OLEUsage`	Устанавливает/возвращает роли OLE-клиента и OLE-сервера, в которых будет использоваться объект `CommandBarControl` при объединении двух приложений Microsoft Office
`OnAction`	`String`	Устанавливает/возвращает имя процедуры Visual Basic, которая запускается при щелчке пользователя или изменении значения объекта `CommandBarControl`
`Parameter`	`String`	Устанавливает/возвращает строку, которая может использоваться приложением при вызове команд
`Priority`	`Long`	Устанавливает/возвращает приоритет объекта `CommandBarControl`
`Tag`	`String`	Устанавливает/возвращает информацию об элементе управления `CommandBarControl`. Например, свойство может содержать данные, которые передаются в процедуры в качестве значений аргументов
`TooltipText`	`String`	Устанавливает/возвращает текст всплывающей подсказки для элемента управления `CommandBarControl`
`Top`	`Long`	Только чтение. Возвращает расстояние (в пикселях) от верхнего края элемента управления `CommandBarControl` до верхнего края экрана
`Type`	`MsoControl-Type`	Только чтение. Возвращает тип элемента управления `CommandBarControl`
`Visible`	`Boolean`	Устанавливает/возвращает видимость элемента управления `CommandBarControl`
`Width`	`Long`	Устанавливает/возвращает ширину (в пикселях) элемента управления `CommandBarControl`

Методы объекта CommandBarControl

Имя	*Тип возвращаемого значения*	*Параметры*	*Описание*
`Copy`	`CommandBar-Control`	`[Bar], [Before]`	Копирует элемент управления `CommandBarControl` на существующую командную панель
`Delete`		`[Temporary]`	Удаляет указанный объект `CommandBarControl` из коллекции. Параметр `Temporary` устанавливается в значение `True` для удаления элемента управления только на время текущего сеанса. После начала следующего сеанса элемент управления опять будет присутствовать на командной панели
`Execute`			Вызывает процедуру или встроенную команду, назначенную указанному элементу управления `CommandBarControl`. Для собственных элементов управления выбор процедуры осуществляется с помощью свойства `OnAction`

Имя	*Тип возвращаемого значения*	*Параметры*	*Описание*
Move	CommandBar-Control	[Bar], [Before]	Перемещает указанный элемент управления CommandBarControl на существующую командную панель
Reset			Сбрасывает встроенный элемент управления CommandBarControl в принятую по умолчанию конфигурацию, восстанавливая функцию и пиктограмму
SetFocus			Перемещает фокус клавиатуры на указанный элемент управления CommandBar-Control. Если элемент управления отключен или не показан на экране, вызов метода завершится ошибкой

Пример: объект CommandBarControl

Следующая подпрограмма выполняет поиск элементов управления CommandBar-Control по значению свойства Tag. В зависимости от типа найденного элемента управления применяются его уникальные свойства. Свойства State и CommandBar, а также методы Clear, которые используются ниже, отсутствуют в списке свойств и методов универсального объекта CommandBarControl, но они будут работать, если элемент управления имеет правильный тип.

```
Sub FindCommandBarControl()

    Dim ctl As CommandBarControl

    ' Найти элемент управления из командной панели Wrox, учитывая
    ' значение свойства Tag.
    ' Параметр Recursive:=True означает, что поиск выполняется и в
    ' подменю (CommandBarPopup)
    Set ctl = CommandBars("Wrox").FindControl(Tag:="MailTo", _
                                              Recursive:=True)

    If Not ctl Is Nothing Then 'We found the control
        ' Получить доступ к уникальному свойству или методу элемента
        ' управления
        Select Case ctl.Type
            Case msoControlButton
                ' Кнопка должна выглядеть нажатой
                ctl.State = msoButtonDown
            Case msoControlComboBox
                ' Очистить список комбинированного списка
                ctl.Clear
            Case msoControlButtonPopup

                ' Получить доступ ко второму элементу управления в
                ' меню/подменю с помощью уникального свойства CommandBar
                ctl.CommandBar.Controls(2).Enabled = False
            Case Else
                MsgBox ctl.Type
        End Select
    End If

End Sub
```

Объект CommandBarPopup

Этот объект описывает меню или подменю на командной панели. Меню и подменю могут содержать вложенные элементы управления командных панелей. Например, меню Файл (File) и Правка (Edit) на панели меню являются элементами управления CommandBarPopup. Подменю Отправить (SendTo) в меню Файл (File) и подменю Заполнение (Fill) в меню Правка (Edit) также являются элементами управления CommandBarPopup.

Так как элементы управления CommandBarPopup могут содержать другие элементы управления, по сути они являются отдельными командными панелями. Если предположить, что первый на собственной командной панели Wrox — элемент управления CommandBarPopup, следующий код можно использовать для обращения к элементу управления и применению его в качестве еще одной командной панели:

```
Dim oBar as CommandBar
Set oBar = CommandBars("Wrox").Controls(1).CommandBar
```

Для обращения к элементу управления в качестве объекта CommandBarPopup необходимо использовать такой код:

```
Dim ctl As CommandBarPopup
Set ctl = CommandBars("Wrox").Controls(1)
```

Общие свойства объекта CommandBarPopup

Определения свойств Application, Creator и Parent приводились в начале приложения.

Свойства объекта CommandBarPopup

Имя	*Тип возвращаемого значения*	*Описание*
BeginGroup	Boolean	Устанавливает/возвращает информацию о том, расположен ли объект CommandBarPopup в начале группы элементов управления на командной панели
BuiltIn	Boolean	Только чтение. Возвращает значение True, если указанная командная панель или элементы управления командной панели являются встроенными. Возвращает значение False, если это собственная командная панель или элемент управления. Кроме того, значение False возвращается для встроенных элементов управления с модифицированным значением свойства OnAction
Caption	String	Устанавливает/возвращает текст заголовка элемента управления CommandBarPopup
CommandBar	CommandBar	Только чтение. Возвращает объект CommandBar, который представляет меню соответствующего всплывающего элемента управления
Controls	CommandBar-Controls	Только чтение. Возвращает коллекцию CommandBarControls, в которой содержатся все элементы управления всплывающего меню
Description-Text	String	Устанавливает/возвращает описание объекта CommandBarPopup. Описание не предоставляется пользователю, но может применяться для документирования поведения элемента управления для других разработчиков

Имя	*Тип возвращаемого значения*	*Описание*
Enabled	Boolean	Устанавливает/возвращает состояние объекта CommandBarPopup (включен/выключен)
Height	Long	Устанавливает/возвращает высоту элемента управления CommandBarPopup
Help-ContextId	Long	Устанавливает/возвращает идентификатор контекста для раздела справочного руководства, связанного с элементом управления CommandBarPopup
HelpFile	String	Устанавливает/возвращает имя файла справочного руководства, связанного с элементом управления CommandBarPopup
Id	Long	Только чтение. Возвращает идентификатор встроенного элемента управления CommandBarPopup
Index	Long	Только чтение. Возвращает число типа Long, являющееся номером объекта CommandBarPopup в коллекции CommandBars
IsPriority-Dropped	Boolean	Только чтение. Сообщает, что объект CommandBarPopup исключен из меню или панели инструментов на основе статистики использования и площади макета. (Обратите внимание, что это свойство не идентично видимости элемента управления, которая устанавливается с помощью свойства Visible.) Если свойство IsPriorityDropped установлено в значение True, установка свойства Visible в значение True не приведет к немедленному появлению элемента управления на экране
Left	Long	Только чтение. Возвращает горизонтальное положение (в пикселях) элемента управления CommandBarPopup относительно левого края экрана. Возвращается расстояние от левого края области присоединения
OLEMenu-Group	MsoOLEMenu-Group	Устанавливает/возвращает группу меню, которой принадлежит объект CommandBarPopup при слиянии групп меню сервера OLE с группами меню клиента OLE. Для встроенных элементов управления это свойство доступно только для чтения
OLEUsage	MsoControl-OLEUsage	Устанавливает/возвращает роли OLE-клиента и OLE-сервера, в которых будет использоваться объект CommandBarPopup при объединении двух приложений Microsoft Office
OnAction	String	Устанавливает/возвращает имя процедуры Visual Basic, которая запускается при щелчке пользователя или изменении значения объекта CommandBarPopup
Parameter	String	Устанавливает/возвращает строку, которая может использоваться приложением при вызове команд
Priority	Long	Устанавливает/возвращает приоритет объекта CommandBarPopup
Tag	String	Устанавливает/возвращает информацию об элементе управления CommandBarPopup, например, свойство может содержать данные, которые передаются в процедуры в качестве значений аргументов

Имя	*Тип возвращаемого значения*	*Описание*
TooltipText	String	Устанавливает/возвращает текст всплывающей подсказки для элемента управления CommandBarPopup
Top	Long	Только чтение. Возвращает расстояние (в пикселях) от верхнего края элемента управления CommandBarPopup до верхнего края экрана
Type	MsoControl-Type	Только чтение. Возвращает тип элемента управления CommandBarPopup
Visible	Boolean	Устанавливает/возвращает видимость элемента управления CommandBarPopup
Width	Long	Устанавливает/возвращает ширину (в пикселях) элемента управления CommandBarPopup

Методы объекта CommandBarPopup

Имя	*Тип возвращаемого значения*	*Параметры*	*Описание*
Copy	CommandBar-Control	[Bar], [Before]	Копирует элемент управления CommandBarPopup в существующую командную панель
Delete		[Temporary]	Удаляет указанный объект CommandBarPopup из коллекции. Параметр Temporary устанавливается в значение True для удаления элемента управления только на время текущего сеанса. После начала следующего сеанса элемент управления опять будет присутствовать на командной панели
Execute			Вызывает процедуру или встроенную команду, назначенную указанному элементу управления CommandBarPopup. Для собственных элементов управления выбор процедуры осуществляется с помощью свойства OnAction
Move	CommandBar-Control	[Bar], [Before]	Перемещает указанный элемент управления CommandBarPopup на существующую командную панель
Reset			Сбрасывает встроенный элемент управления CommandBarPopup в принятую по умолчанию конфигурацию, восстанавливая функцию и пиктограмму
SetFocus			Перемещает фокус клавиатуры на указанный элемент управления CommandBarPopup. Если элемент управления отключен или не показан на экране, вызов метода завершится ошибкой

Пример: объект CommandBarPopup

Следующая подпрограмма добавляет собственное всплывающее меню с тремя элементами управления CommandBarButton. Меню добавляется в существующее меню собственной командной панели Wrox. Информация для добавления трех элементов управления CommandBarButton извлекается из таблицы на листе книги, в которой расположен код.

```
Sub AddCommandBarPopup()

    Dim ctlButton As CommandBarButton
    Dim ctlPopup As CommandBarPopup
    Dim ctlMenuPopup As CommandBarPopup
    Dim rngControls As Range, rngControl As Range

    ' Найти меню Special на командной панели Wrox
    Set ctlMenuPopup = CommandBars("Wrox").FindControl( _
                          Type:=msoControlPopup, _
                          Tag:="PopupSpecial", _
                          Recursive:=True)

    ' Продолжать, если меню найдено
    If Not ctlMenuPopup Is Nothing Then
        ' Добавить элемент управления меню в найденное меню Special,
        ' разместив его на первом месте
        Set ctlPopup = ctlMenuPopup.Controls.Add( _
                          Type:=msoControlPopup, _
                          Before:=ctlMenuPopup.Controls(1).Index)

        ' Выбрать диапазон с таблицей, содержащей описание
        ' добавляемых элементов управления
        With wksPopup
            Set rngControls = .Range(.Range("A2"), _
.Range("A2").End(xlDown))
        End With

        ' Установить заголовок нового пункта меню (с подчеркиванием
        ' буквы E)
        ctlPopup.Caption = "&Edit"

        ' Перебрать таблицу, добавляя элементы управления в новое меню
        For Each rngControl In rngControls
            ' Для встроенных элементов управления
            ' используется параметр ID метода Add
            If Len(rngControl.Offset(0, 3).Value) Then
                Set ctlButton = ctlPopup.Controls.Add( _
                                  Type:=rngControl.Offset(0, 1).Value, _
                                  ID:=rngControl.Offset(0, 3).Value)
            Else
                Set ctlButton = ctlPopup.Controls.Add( _
                                  Type:=rngControl.Offset(0, 1).Value)
            End If
              ' Установить свойства нового элемента управления
            With ctlButton
                ' Если в столбце BeginGroup таблицы есть значение,
                ' добавить разделитель
                .BeginGroup = (Len(rngControl.Offset(0, 2).Value) > 0)
                ' Установить пиктограмму элемента управления,
                ' если в столбце таблицы FaceID присутствует значение.
                ' Для встроенных элементов управления в этом столбце
                ' ничего не должно быть
                If Len(rngControl.Offset(0, 4).Value) Then _
```

```
                .FaceId = rngControl.Offset(0, 4).Value

            ' Назначить макрос собственному элементу управления
            If Len(rngControl.Offset(0, 5).Value) Then _
                .OnAction = rngControl.Offset(0, 5).Text

            ' Установить значение свойства Tag для поиска
            ' элемента управления в будущем
            If Len(rngControl.Offset(0, 6).Value) Then _
                .Tag = rngControl.Offset(0, 6).Text

            ' Использовать текст из столбца ToolTip для установки
            ' свойств Caption и ToolTip
            If Len(rngControl.Offset(0, 7).Value) Then
                .Caption = rngControl.Offset(0, 7).Text
                .TooltipText = rngControl.Offset(0, 7).Text
            End If
        End With
    Next rngControl

    Else
        ' Если меню Special не обнаружено, вывести сообщение
        MsgBox "Невозможно найти элемент управления Member Info",
↳ vbCritical, "Wrox"
    End If

End Sub
```

Ниже показана таблица, на основе которой генерируются три элемента управления `CommandBarButtons`.

Sub Menu	*Type*	*BeginGroup*	*ID*	*Face ID*	*On Action*	*Tag*	*ToolTip Text*
&Select Numeric Values	1			441	Select Numeric Values	Select Numeric Values	Select Numeric Values
Paste Special &Values	1	TRUE	370				
Paste Special &Formulas	1		5836				

Объект DocumentLibraryVersion

Объект `DocumentLibraryVersion` представляет одну сохраненную версию содержащегося в серверной библиотеке документа с поддержкой совместного доступа и включенным управлением версиями. Каждый объект `DocumentLibraryVersion` является членом коллекции `DocumentLibraryVersions`, принадлежащей активному документу.

Свойства объекта DocumentLibraryVersion

Имя	*Тип возвращаемого значения*	*Описание*
`Comments`	`String`	Возвращает необязательные комментарии, связанные с указанной версией документа, поддерживающего совместный доступ

Имя	Тип возвращаемого значения	Описание
Index	Long	Только чтение. Возвращает значение типа Long, описывающее положение объекта в коллекции
Modified	Date/Time	Возвращает дату модификации документа
ModifiedBy	String	Только чтение. Возвращает имя пользователя, который последним сохранял на сервере указанную версию документа, поддерживающего совместный доступ

Методы объекта DocumentLibraryVersion

Имя	Тип возвращаемого значения	Параметры	Описание
Delete			Удаляет объект DocumentLibraryVersion из коллекции DocumentLibraryVersions
Open			Открывает указанную версию поддерживающего совместный доступ документа в режиме только чтения
Restore			Восстанавливает предварительно сохраненную версию поддерживающего совместный доступ документа

Коллекция DocumentLibraryVersions

Хранит список объектов DocumentLibraryVersion.

Коллекция DocumentProperties

Представляет все свойства документа, перечисленные на вкладках Общие (Summary) и Custom в диалоговом окне Свойства (Porperties), доступном по команде меню Файл (File). В Excel документ представлен объектом Workbook. В Word — объектом Document.

В коллекции DocumentProperties хранятся два различных типа объектов. Одни соответствуют встроенным свойствам, другие — собственным. Встроенные свойства характерны для приложения и доступны на вкладке Общие (Summary) диалогового окна Свойства (Properties). Собственые свойства создаются пользователем для определенного документа и предоставляются на вкладке Custom диалогового окна Свойства (Properties).

Важно обратить внимание, что при доступе к коллекции DocumentProperties необходимо использовать свойство BuiltinDocumentProperties для встроенных свойств приложения или свойство CustomDocumentProperties для свойств, созданных пользователем. Странно, но свойства BuiltinDocumentProperties и CustomDocumentProperties не описываются в объектной модели Office, хотя являются частью объектной модели приложения. Другими словами, эти свойства отсутствуют в описании объектной модели Microsoft Office XP.

Для получения доступа к встроенным свойствам документа можно воспользоваться следующим кодом:

```
MsgBox ActiveWorkbook.BuiltinDocumentProperties("Author").Value
```

или

```
MsgBox ActiveWorkbook.BuiltinDocumentProperties(3).Value
```

Стоит знать, что значение Index для свойства Author равно 3. Дополнительная информация приводится ниже в описании коллекции DocumentProperties.

Для доступа к собственным свойствам документа воспользуйтесь следующим кодом:

```
MsgBox ActiveWorkbook.CustomDocumentProperties("BillingNumber").Value
```

или

```
MsgBox ActiveWorkbook.CustomDocumentProperties(1).Value
```

Предполагается, что BillingNumber является первым собственным свойством.

На вкладке Custom в окне Свойства (Properties) в Excel предоставляется несколько вариантов собственных свойств, но в список CustomDocumentProperties попадают только те, которым присвоено значение.

Несколько встроенных свойств используются в конкретных приложениях. Например, свойство NumberofParagraphs используется только в Microsoft Word и любая попытка его применения в других приложениях приводит к появлению ошибки времени выполнения.

Общие свойства коллекции DocumentProperties

Определения свойств Application, Count, Creator и Parent приводились в начале приложения.

Свойства коллекции DocumentProperties

Имя	*Тип возвращаемого значения*	*Параметры*	*Описание*
Item	DocumentProperty	Index As Variant	Возвращает элемент коллекции. Параметр Index может содержать строковое имя объекта DocumentProperty

Методы коллекции DocumentProperties

Имя	*Тип возвращаемого значения*	*Параметры*	*Описание*
Add	DocumentProperty	Name As String, LinkToContent As Boolean, [Type], [Value], [LinkSource]	Создает новое собственное свойство. В коллекцию DocumentProperties можно добавлять только новые собственные свойства документа

Пример: коллекция DocumentProperties

Следующая подпрограмма создает список всех свойств документа. Указывается имя свойства, источник (встроенное или собственное), тип, номер и значение. Номер свойства определяется с помощью переменной counter. Параметр Index определяет положение в списке и может использоваться для доступа к свойству без информации о его имени. Например, в предыдущем коде для встроенного свойства Author использовался номер 3.

```
Sub ListDocumentProperties()

    Dim oProperty As DocumentProperty
    Dim lIndex As Long, lRow As Long

    ' Отключить обновление экрана
```

```
    Application.ScreenUpdating = False

    ' Инициализировать счетчики
    lIndex = 1
    lRow = 2

    ' Получить доступ к листу в пределах книги. Обратите внимание, что
    ' непосредственный доступ к листу
    ' возможен через свойство Name (в окне Свойства (Properties) в редакторе
    ' VBE). Это возможно, если код и лист находятся в пределах одной книги
    With wksBuiltInProperties

        ' Очистить лист кроме заголовков в первой строке
        .UsedRange.Offset(1, 0).ClearContents

        ' Подавить вывод сообщений ошибок при невозможности доступа
        ' к значению свойства из Excel
        On Error Resume Next
         ' Перебрать встроенные свойства
        For Each oProperty In .Parent.BuiltinDocumentProperties
            .Cells(lRow, 1) = oProperty.Name
            .Cells(lRow, 2) = "Встроенное"
            .Cells(lRow, 3) = oProperty.Type
            .Cells(lRow, 4) = lIndex
            .Cells(lRow, 5) = oProperty.Value

            lIndex = lIndex + 1
            lRow = lRow + 1
        Next oProperty

        ' Сбросить значение счетчика Index и перебрать собственные
        ' свойства
        lIndex = 1
        For Each oProperty In .Parent.CustomDocumentProperties
            .Cells(lRow, 1) = oProperty.Name
            .Cells(lRow, 2) = "Собственное"
            .Cells(lRow, 3) = oProperty.Type
            .Cells(lRow, 4) = lIndex
            .Cells(lRow, 5) = oProperty.Value
            lIndex = lIndex + 1
            lRow = lRow + 1
        Next oProperty

        ' Выделить первую ячейку листа
        .Cells(1, 1).Select

    End With

End Sub
```

Объект DocumentProperty

Представляет одно свойство в коллекции `DocumentProperties`. Свойство может быть встроенным или собственным. Для обращения к объекту `DocumentProperty` используются свойства `CustomDocumentProperties` или `BuiltinDocumentProperties`.

Общие свойства объекта DocumentProperty

Определения свойств `Application`, `Creator` и `Parent` приводились в начале приложения.

Свойства объекта DocumentProperty

Имя	*Тип возвращаемого значения*	*Описание*
LinkSource	String	Устанавливает/возвращает источник связанного собственного свойства
LinkToContent	Boolean	Возвращает информацию о том, связано ли собственное свойство документа с содержимым документа
Name	String	Устанавливает/возвращает имя свойства документа
Type	MsoDocProperties	Устанавливает/возвращает тип свойства документа. Для встроенных свойств возможно только чтение. Для собственных свойств возможно чтение и запись
Value	Variant	Устанавливает/возвращает значение свойства документа. Если приложение не определяет значение одного из встроенных свойств документа, чтение значения свойства `Value` для этого документа приводит к появлению сообщения об ошибке

Методы объекта DocumentProperty

Имя	*Тип возвращаемого значения*	*Параметры*	*Описание*
Delete	Result		Удаляет собственное свойство документа

Пример: объект DocumentProperty

Следующая подпрограмма добавляет собственное свойство документа, которое называется `BillingNumber`. Пользователю предоставляется возможность ввести значение свойства `Value`.

```
Sub AddDocumentProperty()

    Dim oProperty As DocumentProperty
    Dim vAnswer As Variant
    Dim wkb As Workbook

    ' Проверить существование активной книги
    On Error Resume Next
        Set wkb = ActiveWorkbook
    On Error GoTo 0

    ' Если книга активна...
    If Not wkb Is Nothing Then

        ' Запросить у пользователя выделение ячейки или ввод числа
        ' Если пользователь выделяет диапазон, может быть
        ' использована только первая ячейка диапазона
        vAnswer = Application.InputBox("Введите значение
↳ BillingNumber?", "Billing Number", , , , , , 1)

        ' Если пользователь не щелкнул на кнопке Отмена (Cancel)...
        If vAnswer <> "False" Then

            ' Проверить существование собственного свойства
            On Error Resume Next
                Set oProperty = _
```

```
ActiveWorkbook.CustomDocumentProperties("BillingNumber")
            On Error GoTo 0

            ' Если свойство не существует, создать его
            If oProperty Is Nothing Then
               Set oProperty = ActiveWorkbook.
↳CustomDocumentProperties.Add (Name:="BillingNumber", _
                               LinkToContent:=False, _
                               Type:=msoPropertyTypeNumber, Value:=1)
            End If

            ' Установить значение свойства Value равным возвращаемому
            ' значению функции InputBox
            oProperty.Value = vAnswer

        End If

    Else
        ' Нет активной книги. Сообщить об этом пользвоателю
        MsgBox "Нет активной книги.", vbCritical, "Добавление
↳ свойства"
    End If

End Sub
```

Объект FileDialog

Этот объект является более структурированной и гибкой альтернативой методам GetSaveAsFilename и GetOpenFilename. Он предоставляет возможность модификации кнопки операции (например кнопки Сохранить (Save) в диалоговом окне Сохранить Как (Save As)) и возможность выбора различных типов диалоговых окон (кроме Открыть (Open) и Сохранить Как (Save As)). Кроме этого, предоставляется большая гибкость при определении собственных типов файлов и фильтров (например, "*.bil") и возможность установки принятого по умолчанию представления, к которому получает доступ пользователь при появлении диалогового окна на экране (например, представления Подробно (Detail) или Крупные значки (Large Icon)).

Обратите внимание, что некоторые свойства и методы этого объекта зависят от значения свойства FileDialogType, которому присваиваются константы MsoFileDialogType. Например, запуск следующего кода приведет к появлению сообщения об ошибке при попытке использования метода Add свойства Filters, если в качестве типа диалогового окна выбран msoFileDialogSaveAs.

```
Application.FileDialog(msoFileDialogSaveAs).Filters.Add _
    "Billing Files", "*.bil", 1
```

Общие свойства объекта FileDialog

Определения свойств Application, Creator и Parent приводились в начале приложения.

Свойства объекта FileDialog

Имя	*Тип возвращаемого значения*	*Описание*
AllowMultiSelect	Boolean	Устанавливает/возвращает возможность выбора нескольких файлов из списка
ButtonName	String	Устанавливает/возвращает текст, который выводится на кнопке операции в диалоговом окне. По умолчанию этому свойству присваивается стандартный текст в соответствии с типом диалогового окна
DialogType	MsoFileDialogType	Только чтение. Возвращает константу MsoFileDialogType, описывающую тип диалогового окна, которое выводится с помощью объекта FileDialog
FilterIndex	Long	Устанавливает/возвращает принятый по умолчанию фильтр для диалогового окна выбора файла. Принятый по умолчанию фильтр определяет тип файлов, которые присутствуют в списке при открытии диалогового окна
Filters	FileDialogFilters	Возвращает коллекцию FileDialogFilters
InitialFileName	String	Устанавливает/возвращает путь и/или имя файла , которые показаны в диалоговом окне в момент открытия
InitialView	MsoFileDialogView	Устанавливает/возвращает константу MsoFileDialogView, описывающую исходное представление файлов и папок в диалоговом окне
Item	String	Только чтение. Возвращает текст, связанный с окном FileDialog
SelectedItems	FileDialogSelectedItems	Возвращает коллекцию FileDialogSelectedItems. В этой коллекции хранится список путей к файлам, которые пользователь выбрал в диалоговом окне, показанном с помощью метода Show объекта FileDialog
Title	String	Устанавливает/возвращает заголовок диалогового окна, созданного с помощью объекта FileDialog

Методы объекта FileDialog

Имя	*Тип возвращаемого значения*	*Параметры*	*Описание*
Execute		For	Выполняет операцию пользователя после вызова метода Show. Используется для объектов FileDialog, свойство Type которых установлено в значение msoFileDialogOpen или msoFileDialogSaveAs
Show	Long		Выводит диалоговое окно на экран. Возвращает значение типа Long, указывающее, щелкнул ли пользователь на кнопке операции (–1) или кнопке отмены (0). При вызове метода Show выполнение кода приостанавливается, пока пользователь не закроет диалоговое окно. Для выполнения пользовательской операции сразу после вызова метода Show диалоговых окон Open и SaveAs необходимо воспользоваться методом Execute

Пример: объект FileDialog

Следующая подпрограмма запрашивает у пользователя разрешение на сохранение активной книги с помощью объекта `FileDialog`. После закрытия диалогового окна Сохранить Как (Save As) подпрограмма выполняет преобразование книги с формулами и именами диапазонов в значения без сохранения имен диапазонов.

```
Sub ConvertBillingStatement()

    ' Воспользуйтесь этой константой для обеспечения однородности в
    ' пределах всей процедуры

    Const sNAME_BILLING As String = "Billing"

    Dim nm As Excel.Name

    Dim sPath As String
    Dim wks As Excel.Worksheet
    Dim wkb As Excel.Workbook

    ' Убедится в том, что это книга учета, проверив существование
    ' скрытого имени
    On Error Resume Next
        Set wkb = ActiveWorkbook
    On Error GoTo 0

    ' Если книга является активной...
    If Not wkb Is Nothing Then
        On Error Resume Next
            Set nm = wkb.Names(sNAME_BILLING)

        ' Если это файл учета...
        If Not nm Is Nothing Then

            ' Сохранить текущий путь
            sPath = CurDir

            ' Установить свойства диалогового окна Сохранить Как
            With Application.FileDialog(msoFileDialogSaveAs)

                ' Изменить принятый по умолчанию заголовок окна
                .Title = "Save " & sNAME_BILLING & " Number"

                ' Изменить текст на кнопке Сохранить (Save) в
                ' пределах диалогового окна
                .ButtonName = "Save " & sNAME_BILLING

                ' Перейти в папку Billing
                .InitialFileName = "C:\" & sNAME_BILLING & "\"

                ' Переключится на представление Подробно (Details)
                .InitialView = msoFileDialogViewDetails

                ' Вывести диалоговое окно. Возвращаемое значение -1
                ' означает, что пользователь не щелкнул на кнопке
                ' Отмена (Cancel)
                If .Show = -1 Then
                    ' Преобразовать все формулы в значения
                    For Each wks In wkb.Worksheets
                        wks.UsedRange.Copy
                        wks.UsedRange.PasteSpecial xlPasteValues
                    Next wks
```

```
                ' Удалить все имена диапазонов кроме имени,
                ' идентифицирующего книгу
                For Each nm In wkb.Names
                    If nm.Name <> sNAME_BILLING Then nm.Delete
                Next nm

                ' Сохранить файл
                .Execute

            End If

            ' Восстановить текущий путь
            ChDir sPath

        End With
    Else
        ' Имя диапазона Billing отсутствует, поэтому это другой
        ' файл
        MsgBox wkb.Name & " is not a " & sNAME_BILLING & "
↳ workbook", _
               vbInformation, "Convert " & sNAME_BILLING & "
↳ Statement"
    End If
  End If

End Sub
```

Коллекция FileDialogFilters

Представляет все фильтры, доступные в диалоговом окне `FileDialog`, включая фильтры, созданные с помощью метода Add свойства `Filters` объекта `FileDialog`.

Обратите внимание, что фильтры, созданные с помощью метода `Add` свойства `Filters`, не отображаются в стандартном диалоговом окне Открыть (Open) и Сохранить Как (Save As).

Общие свойства коллекции FileDialogFilters

Определения свойств `Application`, `Count`, `Creator` и `Parent` приводились в начале этого приложения.

Методы коллекции FileDialogFilters

Имя	*Тип возвращаемого значения*	*Параметры*	*Описание*
`Add`	`FileDialog-Filters`	`Description As String, Extensions As String, [Position]`	Добавляет новый фильтр в список фильтров из раскрывающегося списка Файлы типа (Files of type) в диалоговом окне Файл (File). Возвращает объект `FileDialogFilter`, который соответствует добавленному фильтру
`Clear`			Удаляет все фильтры файлов из коллекции `FileDialogFilters`
`Delete`		`[filter]`	Удаляет указанный фильтр из коллекции `FileDialogFilters`
`Item`	`FileDialog-Filter`	`[Index As Long]`	Возвращает указанный объект `FileDialog-Filter` из коллекции `FileDialogFilters`

Пример: коллекция FileDialogFilters

В следующей подпрограмме объект `FileDialog` используется для вывода диалогового окна **Открыть** (Open). В подпрограмме применяется метод Add свойства `Filters` для добавления фильтра для собственного типа файла. Обратите внимание, что фильтр сохраняется в коллекции и после завершения работы подпрограммы, поэтому подпрограмма удаляет фильтр из списка после закрытия диалогового окна.

```
Sub UsingFileDialogOpen()

    Const lFILTER_POSITION As Long = 1

    Dim lCount As Long
    Dim sChosen As String

    ' Установить свойства диалогового окна Открыть файл (File Open)
    With Application.FileDialog(msoFileDialogOpen)
        ' Изменить принятый по умолчанию заголовок диалогового окна
        .Title = "Открыть файлы учета"

        ' Разрешить пользователю выбирать несколько файлов
        .AllowMultiSelect = True

        ' Установить описание и положение фильтра в списке
        .Filters.Add "Файлы учета", "*.bil", lFILTER_POSITION

        ' Выбрать добавленный фильтр перед выводом диалогового окна
        .FilterIndex = lFILTER_POSITION

        ' Показать диалоговое окно. Значение -1 означает, что пользователь
        ' не щелкнул на кнопке Отмена (Cancel)
        If .Show = -1 Then
            ' Инициализировать строку сообщения
            sChosen = "Были выбраны следующие файлы:" & vbCrLf

            ' Вывести имя файла (и путь) для каждого выбранного файла
            For lCount = 1 To .SelectedItems.Count
                sChosen = sChosen & vbCrLf & .SelectedItems(lCount)
            Next lCount

            ' Показать список выбранных файлов
            MsgBox sChosen, vbInformation

        End If

        ' Удаляет фильтр
        .Filters.Delete lFILTER_POSITION

    End With
End Sub
```

Объект FileDialogFilter

Представляет один фильтр в коллекции `FileDialogFilters`. Для обращения к нему воспользуйтесь следующим кодом:

```
Application.FileDialog(msoFileDialogOpen).Filters(lIndex)
```

Общие свойства объекта FileDialogFilter

Определения свойств Application, Creator и Parent приводились в начале приложения.

Свойства объекта FileDialogFilter

Имя	*Тип возвращаемого значения*	*Описание*
Description	String	Только чтение. Возвращает описание каждого фильтра в диалоговом окне в виде значения типа String
Extenstions	String	Только чтение. Возвращает значение типа String, в котором перечислены расширения, определяющие список файлов в диалоговом окне

Пример: объект FileDialogFilter

В следующей подпрограмме из списка фильтров удаляются все собственные фильтры типов учета из объекта FileDialog типа msoFileDialogSaveAs.

```
Sub RemoveCustomBillingFilters()

    Dim lIndex As Long

    With Application.FileDialog(msoFileDialogOpen)
        ' Перебрать список фильтров в обратном порядке.
        ' Если перебирать список в обычном порядке, удаление фильтров
        ' приведет к смещению остальных фильтров вверх на одну
        ' позицию.
        ' При этом цикл пропускает фильтры после удаленных фильтров.
        For lIndex = .Filters.Count To 1 Step -1
            ' Если расширение содержит символы "bi", это собственный
            ' фильтр и его можно удалить.
            If .Filters(lIndex).Extensions Like "*.bi*" Then
                .Filters.Delete lIndex
            End If
        Next lIndex
    End With

End Sub
```

Коллекция FileDialogSelectedItems

Эта коллекция содержит выбранные элементы из диалогового окна, доступного через объект FileDialog. Коллекция будет содержать больше одного элемента, если установить в значение True свойство AllowMultiSelect объекта FileDialog. При использовании диалогового окна типа msoFileDialogSaveAs коллекция всегда содержит один элемент. В коллекции FileDialogSelectedItems хранятся строки.

Общие свойства коллекции FileDialogSelectedItems

Определения свойств Application, Count, Creator и Parent приводились в начале приложения.

Методы коллекции FileDialogSelectedItems

Имя	*Тип возвращаемого значения*	*Параметры*	*Описание*
`Item`	`String`	`Index As Long`	Возвращает путь к одному из выделенных пользователем файлов. Список файлов предоставляется пользователю с помощью диалогового окна, которое выводится с помощью метода `Show` объекта `FileDialog`

Пример: коллекция FileDialogSelectedItems

Следующая подпрограмма является модификацией процедуры `ConvertBillingStatement`. В этой версии для получения пути и имени файла из диалогового окна типа `msoFileDialogSaveAs` используется коллекция `SelectedItems`. После этого возвращаемое значение проверяется на наличие фразы "Wrox Billing". В случае ее отсутствия диалоговое окно `FileDialog` выводится повторно.

```
Sub ConvertBillingStatement2()

    ' Эта константа используется для обеспечения одинаковости
    ' ключевого слова в пределах процедуры
    Const sNAME_BILLING As String = "Billing"

    Dim bBillName As Boolean
    Dim nm As Excel.Name
    Dim sPath As String
    Dim wks As Excel.Worksheet
    Dim wkb As Excel.Workbook

    ' Убедится, что это книга учета, проверив существование скрытого
    ' имени
    On Error Resume Next
        Set wkb = ActiveWorkbook
    On Error GoTo 0

    ' Если книга активна...
    If Not wkb Is Nothing Then
        On Error Resume Next
            Set nm = wkb.Names(sNAME_BILLING)
          ' Если это книга учета...
        If Not nm Is Nothing Then

            ' Сохранить текущий путь
            sPath = CurDir

            ' Установить свойства диалогового окна Сохранить как
            With Application.FileDialog(msoFileDialogSaveAs)

                ' Изменить принятый по умолчанию заголовок диалогового
                ' окна
                .Title = "Save " & sNAME_BILLING & " Number"

                '  Изменить надпись на кнопке Сохранить
                .ButtonName = "Сохранить " & sNAME_BILLING

                ' Перейти в папку Billing
                .InitialFileName = "C:\" & sNAME_BILLING & "\"
                .AllowMultiSelect = True

                ' Переключить диалоговое окно на представление Подробности
```

```
        .InitialView = msoFileDialogViewDetails

        ' Выполнять цикл, пока путь или имя файла не будут
        ' содержать фразу "Wrox Billing" (или пользователь
        ' не щелкнет на кнопке Отмена (Cancel)
        Do
           ' Предположить, что путь или имя файла содержат фразу
           ' "Wrox Billing". Для этого установить переменную
           ' BillName в значение True
           bBillName = True

           ' Вывести диалоговое окно. Значение -1 означает, что
           ' пользователь не щелкнул на кнопке Отмена (Cancel)
           If .Show = -1 Then

              ' Настроить проверку, выполняя поиск фразы "Wrox
              ' Billing" в имени файла или каталога
              bBillName = (InStr(1, .SelectedItems(1), _
                          "Wrox " & sNAME_BILLING, vbTextCompare) > 0)

              ' Если фраза найдена...
              If bBillName Then
                 ' Выполнить преобразование формул в значения
                 For Each wks In wkb.Worksheets
                    wks.UsedRange.Copy
                    wks.UsedRange.PasteSpecial xlPasteValues
                 Next wks

                 ' Удалить все имена диапазонов кроме имени,
                 ' идентифицирующих книгу учета
                 For Each nm In wkb.Names
                    If nm.Name <> sNAME_BILLING Then nm.Delete
                 Next nm

                 ' Сохранить файл
                 .Execute

              Else
                 ' Предупредить пользователя о необходимости
                 ' фразы "Wrox Billing" в имени файла или каталога
                 MsgBox "Имя файла должно содержать фразу Wrox
↳ Billing", vbExclamation, "Преобразование книги"
              End If
           End If
        Loop Until bBillName

        ' Вернуть исходный путь
        ChDir sPath

        End With
    Else
        ' Имя диапазона Billing отсутствует, поэтому это не файл
        ' учета
        MsgBox wkb.Name & " отсутствует в книге " & sNAME_BILLING,
↳ vbInformation, "Convert " & sNAME_BILLING & " Statement"
    End If
  End If

End Sub
```

Объект FileSearch

Объект FileSearch программно имитирует возможность поиска из диалогового окна Открыть (Open) (команда Сервис⇨Поиск (Tools⇨Search)). Эта возможность позволяет искать файлы всех типов в каждой группе папок на основе любого набора критериев. Обратите внимание, что параметры поиска сохраняются между обращениями к объекту и должны сбрасываться с помощью метода NewSearch при каждом использовании объекта FileSearch. Кроме этого, обратите внимание, что метод NewSearch не сбрасывает значение свойства LookIn.

Общие свойства объекта FileSearch

Определения свойств Application, Creator и Parent приводились в начале приложения.

Свойства объекта FileSearch

Имя	*Тип возвращаемого значения*	*Описание*
FileName	String	Устанавливает/возвращает имя искомого файла. Имя может содержать символы шаблонов * или ?
FileType	MsoFileType	Устанавливает/возвращает тип искомого файла
FileTypes	FileTypes	Возвращает коллекцию FileTypes
FoundFiles	FoundFiles	Только чтение. Возвращает объект FoundFiles, содержащий имена всех найденных файлов
LastModified	MsoLast-Modified	Устанавливает/возвращает константу, которая определяет время с момента последней модификации и сохранения файла
LookIn	String	Устанавливает/возвращает папку, в которой будет выполняться поиск файлов
MatchAll-WordForms	Boolean	Устанавливает/возвращает необходимость расширения поиска с использованием всех словоформ ключевого слова в содержимом файла или в его свойствах
MatchText-Exactly	Boolean	Устанавливает/возвращает необходимость поиска только тех файлов, в содержимом или свойствах которых присутствуют точное слово или фраза
Property-Tests	PropertyTests	Только чтение. Возвращает коллекцию PropertyTests, в которой хранятся все критерии поиска
Search-Folders	SearchFolders	Возвращает коллекцию SearchFolders
Search-Scopes	SearchScopes	Возвращает коллекцию SearchScopes
SearchSub-Folders	Boolean	Устанавливает/возвращает необходимость поиска в подпапках текущей папки, указанной с помощью свойства LookIn
TextOr-Property	String	Устанавливает/возвращает слово или фразу, которую необходимо найти в теле или в свойствах файла. Слово или фраза могут содержать символы шаблона * или ?

Методы объекта FileSearch

Имя	*Тип возвращаемого значения*	*Параметры*	*Описание*
Execute	Long	[SortBy], [SortOrder], [AlwaysAccurate]	Начинает поиск указанного файла или файлов. Возвращает значение 0, если ни один файл не найден. Положительное возвращаемое значение указывает, что найден один или больше файлов
NewSearch			Сбрасывает критерии поиска в принятые по умолчанию значения
RefreshScopes			Обновляет список доступных объектов ScopeFolder

Пример: объект FileSearch

В следующей подпрограмме выполняется поиск всех файлов, содержащих слово "billing" в папке с файлами учета. Поиск выполняется среди файлов, которые были модифицированы в течение последних семи дней. Результат выводится в окне сообщения.

```
Sub SearchForRecentBilling()

    Dim lCount As Long
    Dim sFiles As String

    With Application.FileSearch

        ' Сбрасывает предыдущие параметры поиска
        .NewSearch

        ' Искать файлы .xls, содержащие слово "Billing"
        .Filename = "*Billing*.xls"

        ' Искать файлы, модифицированные в течение последней недели
        .LastModified = msoLastModifiedThisWeek

        ' Искать в папке и подпапках
        .LookIn = "C:\Billing"
        .SearchSubFolders = True

        ' Начать поиск и вернуть результат
        If .Execute > 0 Then
            sFiles = "Файлов найдено:" & vbCrLf
            For lCount = 1 To .FoundFiles.Count
                sFiles = sFiles & vbCrLf & .FoundFiles(lCount)
            Next lCount
            MsgBox sFiles, vbInformation, "Файлы учета"
        End If

    End With

End Sub
```

Коллекция FileTypes

Представляет набор типов файлов, поиск которых осуществляется с помощью объекта FileSearch. Типы файлов в этой коллекции сохраняются между операциями поиска, поэтому при поиске файлов другого типа важно удалить все типы файлов из коллекции FileTypes. Для этого используется метод Remove объекта коллекции. Новый тип файла устанавливается с помощью свойства FileType.

Общие свойства коллекции FileTypes

Определения свойств Application, Count и Creator приводились в начале приложения.

Свойства коллекции FileTypes

Имя	*Тип возвращаемого значения*	*Параметры*	*Описание*
Item	MsoFileType	Index As Long	Только чтение. Возвращает значение, определяющее тип искомого файла

Методы коллекции FileTypes

Имя	*Тип возвращаемого значения*	*Параметры*	*Описание*
Add		FileType As MsoFileType	Добавляет новый тип файла в критерии поиска
Remove		Index As Long	Удаляет указанный тип файла из коллекции FileTypes

Пример: коллекция FileTypes

Следующая функция сбрасывает объект FileSearch, вызывая метод NewSearch. Свойство LookIn сбрасывается в принятый по умолчанию путь C:\. Для удаления объектов FileType из коллекции FileTypes используется цикл, перебирающий элементы коллекции в обратном порядке.

```
Sub NewSearch()

    ' Сбросить объект FileSearch
    ResetFileSearch

    With Application.FileSearch
        ' Здесь необходимо добавить код для определения параметров
        ' поиска
    End With

End Sub

Function ResetFileSearch()

    Dim lCount As Long
    Dim oFileTypes As FileTypes
    Dim oPropertyTests As PropertyTests

    With Application.FileSearch
```

```
        ' Очистить предыдущие параметры поиска
        .NewSearch

        ' Сбросить свойство LookIn, установоив его в принятое по
        ' умолчанию значение
        ' Вызов метода NewSearch не приводит к сбросу этого свойства
        .LookIn = "C:\"

        ' Удалить все объекты FileType из коллекции FileTypes
        Set oFileTypes = .FileTypes

        ' При удалении содержимого коллекции FileTypes, PropertyTests
        ' и SearchFolders номер элемента после удаляемого уменьшается на
        ' единицу.
        ' Поэтому коллекцию необходимо перебирать в обратном порядке
        ' для избежания сообщения об ошибке Subscript Out of Range
        For lCount = oFileTypes.Count To 1 Step -1
            oFileTypes.Remove lCount
        Next lCount

        ' Очистить коллекцию PropertyTests
        Set oPropertyTests = .PropertyTests
        For lCount = oPropertyTests.Count To 1 Step -1
            oPropertyTests.Remove lCount
        Next lCount

        ' Очистить коллекцию SearchFolders
        For lCount = .SearchFolders.Count To 1 Step -1
            .SearchFolders.Remove lCount
        Next lCount

    End With

End Function
```

Объект FoundFiles

Объект FoundFiles содержит список файлов, найденных в результате поиска.

Общие свойства объекта FoundFiles

Определения свойств Application, Creator и Parent приводились в начале приложения.

Свойства объекта FileTypes

Имя	*Тип возвращаемого значения*	*Параметры*	*Описание*
Item	String	Index As Long	Только чтение. Возвращает имя файла из списка

Объект HTMLProject

Этот объект представляет код HTML, который используется для вывода документа Office в виде документа HTML. Для доступа к HTML-версии документа Office можно воспользоваться редактором Microsoft Script Editor.

Информация об управлении объектами и параметрами с помощью Microsoft Script Editor выходит за пределы тематики этой книги. Но здесь приводится краткое описание перед использованием объектов HTML из объектной модели Office.

Редактор Script Editor похож на редактор Visual Basic Editor. В нем доступно окно Project Explorer (Окно проекта), окно Properties (Свойства), панель Toolbox (Элементы управления) для добавления элементов управления и окно Code (Код) для создания кода HTML. Модификация параметров и содержимого документа в редакторе Script Editor приводит к модификации документа Office. В Excel данные выводятся в формате электронной таблицы с использованием технологии WYSIWYG (What You See Is What You Get), то есть представление на экране совпадает с конечным результатом. Редактор Script Editor предоставляет доступ ко всем элементам форматирования, параметрам, данным и коду HTML в одном окне. Форматирование не выводится, но предоставляется доступ к коду HTML, составляющему форматирование.

Редактор Script Editor позволяет модифицировать HTML-версию документа Office через добавление и модификацию кода HTML, изменение параметров объектов, добавление событий и т.д. Редактирование параметров и кода HTML в редакторе Script Editor чаще всего приводит к изменению документа Office.

Объект HTMLProject содержит объекты так же, как объект VBProject хранит объекты книги. Например, в объекте VBProject для книги хранятся объекты `Sheet` для каждого листа в ее пределах. Редактор Script Editor позволяет редактировать свойства объекта HTML в окне Properties (Свойства), похожее на окно Properties (Свойства) в редакторе VBE, хотя большинство свойств не совпадают со свойствами в редакторе VBE. Например, в редакторе Script Editor можно выполнить двойной щелчок на объекте `Sheet` и в окне Properties (Свойства) изменить значение свойства `Background`, указывая на рисунок, хранящийся на жестком диске. Если проект сохранить на диске и открыть в окне Excel, фоновый рисунок будет показан на экране. В редакторе VBE такого свойства объекта `Sheet` не существует, но для доступа к этому параметру можно воспользоваться командой меню Excel Формат⇨Лист⇨Подложка (Format⇨Sheet⇨Background). Кроме того, свойство можно изменить программным образом из кода VBA.

Объекты HTML в объектной модели Office поддерживают доступ и управление кодом HTML документа Office. Это похоже на возможности объектной модели VBE по предоставлению доступа к VBProject и его объектам.

Общие свойства объекта HTMLProject

Определения свойств `Application`, `Creator` и `Parent` приводились в начале приложения.

Свойства объекта HTMLProject

Имя	*Тип возвращаемого значения*	*Описание*
`HTMLProjectItems`	`HTMLProjectItems`	Только чтение. Возвращает коллекцию `HTMLProjectItems`, которая включена в указанный проект HTML
`State`	`MsoHTMLProjectState`	Только чтение. Возвращает текущее состояние объекта `HTMLProject`

Методы объекта HTMLProject

Имя	*Тип возвращаемого значения*	*Параметры*	*Описание*
Open		[OpenKind As MsoHTMLProjectOpen]	Открывает указанный проект HTML в редакторе Microsoft Script Editor с использованием одного из представлений, выбранных с помощью константы MsoHTMLProjectOpen. Если не указана ни одна из констант, проект открывается с использованием принятого по умолчанию представления
Refresh-Document		Refresh As Boolean = True	Обновляет указанный проект HTML в приложении Microsoft Office. При установке параметра в значение True все внесенные изменения сохраняются. При установке параметра в значение False все изменения игнорируются
Refresh-Project		Refresh As Boolean = True	Использование этого метода эквивалентно щелчку на кнопке Refresh на панели инструментов Refresh в редакторе Microsoft Script Editor. Если при обновлении документа параметр Refresh установить в значение True, все изменения в исходном коде HTML будут сохранены в проекте HTML в редакторе Microsoft Script Editor. Если параметр установить в значение False, все изменения кода HTML будут проигнорированы

Пример: объект HTMLProject

Следующая подпрограмма устанавливает ссылку на объект HTMLProject из активной книги и использует эту ссылку для предоставления количества элементов и состояния блокировки проекта. После этого проект выводится в редакторе Script Editor.

```
Sub HTMLProjectInfo()

    Dim oHTMLProject As HTMLProject

    ' Сохранить ссылку на проект HTML
    Set oHTMLProject = ActiveWorkbook.HTMLProject
    With oHTMLProject
        ' Вывести количество элементов проекта и состояние блокировки
        ' документа или проекта
        MsgBox "Количество элементов проекта: " & _
.HTMLProjectItems.Count & vbCrLf & _
               "Состояние блокировки: " & .State, vbInformation, _
               "Информация о проекте HTML"

        ' Показать проект в редакторе Script Editor
        .Open (msoHTMLProjectOpenSourceView)
    End With

End Sub
```

Коллекция HTMLProjectItems

Представляет все объекты (элементы) проекта HTML. Проекты содержат объект StyleSheet, в котором хранятся стили для всего проекта (это напоминает возможность стилевой разметки в Excel и Word). Все проекты содержат объект, представляющий документ (книгу).

Например, книга Wrox Examples.xls будет содержать элемент HTMLProjectItem, который называется WroxExamples.xls. Этот объект будет храниться в коллекции HTMLProjectItems, принадлежащей объекту HTMLProject из этой книги. В этом элементе проекта хранится информация о текущем приложении, объектах внутри проекта (например, листах книги), а также о свойствах документа (встроенных и собственных; такие свойства рассматривались ранее в разделе, посвященном коллекции DocumentProperties). Кроме этого, каждый лист в Excel является объектом из коллекции HTMLProjectItems.

Общие свойства коллекции HTMLProjectItems

Определения свойств Application, Count, Creator и Parent приводились в начале приложения.

Методы коллекции HTMLProjectItems

Имя	*Тип возвращаемого значения*	*Параметры*	*Описание*
Item	HTMLProjectItem	Index As Variant	Возвращает объект HTMLProjectItem, представляющий определенный проект в редакторе Microsoft Script Editor

Объект HTMLProjectItem

Представляет один элемент коллекции HTMLProjectItems.

Общие свойства объекта HTMLProjectItem

Имя	*Тип возвращаемого значения*	*Описание*
IsOpen	Boolean	Только чтение. Возвращает информацию о том, открыт ли указанный элемент проекта HTML в редакторе Microsoft Script Editor
Name	String	Только чтение. Возвращает имя объекта HTMLProjectItem
Text	String	Устанавливает/возвращает текст кода HTML в редакторе HTML

Методы объекта HTMLProjectItem

Имя	*Тип возвращаемого значения*	*Параметры*	*Описание*
LoadFromFile		FileName As String	Обновляет текст в редакторе Microsoft Script Editor на основе содержимого файла

Имя	*Тип возвращаемого значения*	*Параметры*	*Описание*
Open		[OpenKind As MsoHTMLProjectOpen]	Открывает указанный элемент проекта HTML в редакторе Microsoft Script Editor с использованием одного из указанных с помощью констант MsoHTMLProjectOpen представлений. Если значение параметра не указано, элемент проекта открывается с использованием принятого по умолчанию представления
Save-CopyAs		FileName As String	Сохраняет указанный элемент проекта HTML в файле с новым именем

Пример: объект HTMLProjectItem

Следующая подпрограмма сохраняет код HTML для элемента Workbook из коллекции HTMLProjectItems в текстовом файле и выводит код в редакторе Script Editor.

```
Sub OpenAndExportWorkbookItem()

    Dim oHTMLItem As HTMLProjectItem

    ' Сохранить ссылку на объект Workbook для этого проекта
    ' Обратите внимание: используйте 1 при запуске этого кода в
    ' многоязыковой среде
    ' 1 всегда является объектом Workbook (Document) из проекта
    ' HTMLProject
    Set oHTMLItem = ThisWorkbook.HTMLProject.HTMLProjectItems
↳("WroxExamples.xls")

    With oHTMLItem
        ' Сохранить код HTML для элемента Workbook
        .SaveCopyAs "C:\Billing\WorkbookSettings.txt"

        ' Переключиться на редактор Script Editor
        .Open
    End With

End Sub
```

Объект LanguageSettings

Возвращает информацию о языковых параметрах, которые в данный момент используются в приложении. Эти параметры доступны только для чтения и управляют отображением и редактированием данных в некоторых приложениях.

Общие свойства объекта LanguageSettings

Определения свойств Application, Creator и Parent приводились в начале приложения.

Свойства объекта LanguageSettings

Имя	*Тип возвращаемого значения*	*Описание*
`LanguageID`	`Long`	Только чтение. Возвращает идентификатор региональных параметров (LCID) для языков программы установки, пользовательского интерфейса или справочного руководства
`LanguagePreferredForEditing`	`Boolean`	Только чтение. Возвращает значение `True`, если значение константы `msoLanguageID` указано в системном реестре Windows в качестве идентификатора предпочтительного языка редактирования

Пример: объект LanguageSettings

Следующая подпрограмма выводит сообщение при недоступности дополнительных языковых режимов. В Access и Excel дополнительные языковые режимы могут оказывать влияние на внешний вид и поведение приложения.

```
Sub ChangeLanguageID()

    Dim oLanguage As LanguageSettings

    ' Получить ссылку на объект LanguageSettings
    Set oLanguage = Application.LanguageSettings

    ' Проверить наличие поддержки дополнительных языков и вывести
    ' результат при их отсутствии
    If oLanguage.LanguageID(msoLanguageIDExeMode) = 0 Then
     MsgBox "Поддержка многоязыкового просмотра и редактирования" & _
         "отсутствует. Установите дополнительные языковые пакеты" & _
         "Дополнительная информация приводится в справочном
↳ руководстве в разделе поиска" & _
                "идентификатора LanguageID.", _
                vbInformation, "Language ID"
    End If

End Sub
```

Объект MsoEnvelope

Это новый объект, позволяющий отправлять данные из приложения с помощью почтового агента Outlook без использования объектной модели Outlook. Свойство `Index` этого объекта обеспечивает доступ к возможностям агента Outlook, недоступным через функцию `SendMail`, например, к полям Voting Options, CC и BCC. Кроме этого, предоставляется доступ к параметрам форматирования тела сообщения (HTML, Richtext и простой текст) и другим параметрам сообщения.

Обратите внимание, что объект `MsoEnvelope` отправляет документ внутри тела сообщения в виде отформатированного текста, а не присоединяет документ к сообщению. При этом существует возможность создания вложений с помощью свойства `Attachments` объекта `MaiItem`, доступного через свойство `Index`. В Excel этот объект доступен только через объекты `Worksheet` или `Chart`, а значит отправляются только эти объекты, а не вся книга. Данный объект напоминает функцию `SendMail` в Excel, но он предоставляет объект `CommandBar` и поддерживает добавление текста вступления (Introduction).

Установленные свойства сохраняются в документе/книге и используются в дальнейшем.

Общие свойства объекта MsoEnvelope

Определение свойства `Parent` приводилось в начале приложения.

Свойства объекта MsoEnvelope

Имя	*Тип возвращаемого значения*	*Описание*
`CommandBars`	`CommandBars`	Возвращает коллекцию `CommandBars`
`Introduction`	`String`	Устанавливает/возвращает вступительный текст, который отправляется вместе с документом с помощью объекта `MsoEnvelope`. Вступительный текст добавляется в начало документа в сообщении электронной почты
`Item`	`Object`	Возвращает объект `MailItem`, который может использоваться для отправки документа в виде сообщения электронной почты

События объекта MsoEnvelope

Имя	*Параметры*	*Описание*
`EnvelopeHide`		Возникает при сокрытии пользовательского интерфейса, связанного с объектом `MsoEnvelope`
`EnvelopeShow`		Возникает при отображении пользовательского интерфейса, связанного с объектом `MsoEnvelope`

Пример: объект MsoEnvelope

Следующая подпрограмма создает сообщение (Outlook MailItem) для листа (для этого используется свойство `Name` из окна Properties (Свойства) в редакторе VBE) и устанавливает тему, вступительный текст и получателя. После этого вся книга добавляется в качестве включения и подпрограмма отправляет сообщение.

```
Sub SendSheet()
    SendMsoMail "Robert Rosenberg"
End Sub

Function SendMsoMail(ByVal strRecipient As String)

    Dim oMailEnv As MsoEnvelope
    Dim oMailItem As MailItem

    ' Получить ссылку на объект MsoEnvelope
    Set oMailEnv = wksIncome.MailEnvelope

    ' Настроить конверт
    With oMailEnv

        ' Добавить вступительный текст, который добавляется над
        ' фактическими данными в сообщении Outlook MailItem
        .Introduction = "Вот цифры, которые Вы просили." & _
             "Для удобства в сообщение включена вся книга."

        ' Получить ссылку на объект MailItem, поддерживающий доступ к
        ' свойствам и методам Outlook MailItem
        Set oMailItem = .Item

        ' Настроить объект MailItem
```

```
        With oMailItem

            ' Включить книгу в сообщение
            .Attachments.Add ThisWorkbook.FullName

            ' Выбрать форматирование с помощью HTML
            .BodyFormat = olFormatHTML

            ' Добавить имя получателя и проверить его с помощью
            ' функции Outlook Check Name
            .Recipients.Add strRecipient
            .Recipients.ResolveAll

            ' Добавить тему сообщения
            .Subject = "Вот документ."

            ' Отправить сообщение. Метод Display не работает.
            .Send

        End With

    End With

End Function
```

Объект NewFile

Соответствует новому документу в панели задач приложения. В Excel этот объект позволяет добавлять книги в любой из пяти разделов области задач: Открыть книгу, Чистая книга, На основе имеющейся книги, На основе шаблона или в нижний раздел (без имени). При щелчке на добавленной книге в разделах Чистая книга, Из имеющейся книги или На основе шаблона Excel по умолчанию создаст копию файла, если не переопределить это поведение с помощью параметра Action метода Add.

Общие свойства объекта NewFile

Определения свойств Application и Creator приводились в начале приложения.

Методы объекта NewFile

Имя	*Тип возвращаемого значения*	*Параметры*	*Описание*
Add	Boolean	FileName As String, [Section], [DisplayName], [Action]	Добавляет новый элемент в область задач
Remove	Boolean	FileName As String, [Section], [DisplayName], [Action]	Удаляет элемент из области задач

Пример: объект NewFile

Следующая подпрограмма добавляет новый файл в раздел Из имеющейся книги. В области задач этот пункт будет называться Новая книга учета, но при щелчке будет выполняться копирование книги NewFile.xls, расположенной на жестком диске в каталоге Billing.

```
Sub AddNewWorkbookToTaskPane()

    Dim oNewFile As NewFile

    ' Получить ссылку на объект NewFile
    Set oNewFile = Application.NewWorkbook

    ' Добавить файл в область задач. Элемент добавляется в раздел из
    ' существующей книги
    oNewFile.Add Filename:="C:\Billing\NewFile.xls", _
                 Section:=msoNewfromExistingFile, _
                 DisplayName:="Новая книга учета", _
                 Action:=msoCreateNewFile

    ' Новый элемент области задач не появится на экране, пока область
    ' задач не будет сокрыта и показана повторно
    Application.CommandBars("Task Pane").Visible = False
    Application.CommandBars("Task Pane").Visible = True

End Sub
```

Коллекция ODSOColumns

Соответствует набору полей данных (столбцов) из источника данных Mail Merge.

> На данный момент не может быть реализована. Требует ссылки на объект OfficeDataSourceObject через объект Application текущего приложения. В объектной модели Office XP объект OfficeDataSourceObject отсутствует*.

Общие свойства коллекции ODSOColumns

Определения свойств Application, Count, Creator и Parent приводились в начале приложения.

Методы коллекции ODSOColumns

Имя	*Тип возвращаемого значения*	*Параметры*	*Описание*
Item	ODSOColumn	Index As Integer	Возвращает объект ODSOColumn из коллекции

Объект ODSOColumn

Соответствует одному полю источника данных Mail Merge.

> На данный момент не может быть реализован. Требует ссылки на объект OfficeDataSourceObject через объект Application текущего приложения. В объектной модели Office XP объект OfficeDataSourceObject отсутствует.

Общие свойства объекта ODSOColumn

Определения свойств Application, Creator и Parent приводились в начале приложения.

---

* *В объектной модели Office 2003 этот объект появился.* — Примеч. ред.

Свойства объекта ODSOColumn

Имя	*Тип возвращаемого значения*	*Описание*
`Index`	`Long`	Только чтение. Возвращает положение объекта в коллекции
`Name`	`String`	Устанавливает/возвращает имя текущего объекта
`Value`	`String`	Устанавливает/возвращает значение текущего объекта

Коллекция ODSOFilters

Представляет набор фильтров, применяемых к источнику данных Mail Merge. Фактически фильтры являются запросами, ограничивающими множество возвращаемых записей при использовании источника данных Mail Merge.

> На данный момент не может быть реализована. Требует ссылки на объект `OfficeDataSourceObject` через объект `Application` текущего приложения. В объектной модели Office XP объект `OfficeDataSourceObject` отсутствует.

Общие свойства коллекции ODSOFilters

Определения свойств `Application`, `Count`, `Creator` и `Parent` приводились в начале приложения.

Методы коллекции ODSOFilters

Имя	*Тип возвращаемого значения*	*Параметры*	*Описание*
`Add`			Добавляет новый фильтр в коллекцию. Дополнительная информация доступна в справочном руководстве
`Remove`			Удаляет фильтр из коллекции. Дополнительная информация доступна в справочном руководстве
`Item`			Возвращает объект `ODSOFilter` из коллекции

Объект ODSOFilter

Соответствует одному фильтру из коллекции `ODSOFilters`.

> На данный момент не может быть реализован. Требует ссылки на объект `OfficeDataSourceObject` через объект `Application` текущего приложения. В объектной модели Office XP объект `OfficeDataSourceObject` отсутствует.

Общие свойства объекта ODSOFilter

Определения свойств `Application`, `Creator` и `Parent` приводились в начале приложения.

Свойства объекта ODSOFilter

Имя	*Тип возвращаемого значения*	*Описание*
Column	String	Устанавливает/возвращает имя поля в источнике данных Mail Merge, для которого используется фильтр
CompareTo	String	Устанавливает/возвращает текст для сравнения в критерии фильтра
Conjunction	MsoFilterConjunction	Устанавливает/возвращает константу MsoFilterConjunction, определяющую соединение фильтра с другими фильтрами из коллекции ODSOFilters
Index	Long	Только чтение. Возвращает положение объекта в коллекции

Объект OfficeDataSourceObject

Представляет источник данных при операциях Mail Merge. Позволяет получать набор записей, которые удовлетворяют определенному критерию.

> На данный момент не может быть реализован. Требует ссылки на объект OfficeDataSourceObject через объект Application текущего приложения. В объектной модели Office XP объект OfficeDataSourceObject отсутствует.

Свойства объекта OfficeDataSourceObject

Имя	*Тип возвращаемого значения*	*Описание*
Columns	ODSOColumns	Возвращает коллекцию ODSOColumns, предоставляющую доступ к полям источника данных
ConnectString	String	Устанавливает/возвращает строку подключения к указанному источнику данных Mail Merge
DataSource	String	Устанавливает/возвращает имя связанного источника данных
RowCount	Long	Только чтение. Возвращает количество записей в указанном источнике данных
Table	String	Только чтение. Возвращает имя таблицы из источника данных, содержащей записи Mail Merge

Объект Permission

Объект Permission используется для ограничения доступа к активному документу, а также для получения или установки определенных параметров этого ограничения.

Свойства объекта Permission

Имя	*Тип возвращаемого значения*	*Описание*
Application	Object	Возвращает объект Application, соответствующий приложению, в котором хранится текущий объект
Count	Long	Устанавливает/возвращает значение типа Long, описывающее количество элементов в коллекции

Имя	*Тип возвращаемого значения*	*Описание*
Creator	Long	Возвращает значение типа Long, указывающее на приложение, создавшее текущий объект
DocumentAuthor	String	Устанавливает/возвращает имя автора активного документа
Enabled	Boolean	Устанавливает/возвращает состояние ограничения доступа для активного документа (включено/выключено)
EnableTrustedBrowser	Boolean	Определяет возможность просмотра документа в Web-обозревателе с ограниченными разрешениями, если приложение не установлено
Item	UserPermission	Возвращает объект UserPermission, входящий в коллекцию Permission. Объект UserPermission связывает набор разрешений доступа к активному документу с одним пользователем и датой истечения действительности разрешений на доступ
Parent	Object	Возвращает ссылку на родительский объект текущего объекта
PermissionFromPolicy	Boolean	Возвращает значение типа Boolean, указывающее на применение разрешений доступа к активному документу
PolicyDescription	String	Возвращает описание политики разрешений, применяемой к активному документу
PolicyName	String	Возвращает имя политики разрешений, применяемой к активному документу
RequestPermissionURL	String	Возвращает Web-адрес или адрес электронной почты ответственного лица, к которому можно обратиться для получения разрешения на доступ к документу
StoreLicenses	Boolean	Возвращает значение Boolean, указывающее на необходимость кэширования пользовательской лицензии для поддержки автономного просмотра документа при невозможности подключения к серверу управления правами доступа

Методы объекта Permission

Имя	*Тип возвращаемого значения*	*Параметры*	*Описание*
Add	UserPermission	[UserID], [Permission], [ExpirationDate]	Создает новый набор разрешений доступа к активному документу для указанного пользователя
ApplyPolicy		[Filename]	Применяет указанную политику к активному документу
RemoveAll			Удаляет все объекты UserPermission из коллекции Permission активного документа

Коллекция PropertyTests

Эта коллекция содержит список критериев поиска, которые используются объектом FileSearch. Коллекция аналогична вкладке Расширенный (Advanced) диалогового окна Поиск файлов (Search). Для доступа к этому диалоговому окну выберите команду

Сервис⇨Найти (Tools⇨Search) в диалоговом окне Файл⇨Открыть (File⇨Open). Данная коллекция используется для установки нескольких критериев поиска файлов.

Для добавления расширенных критериев поиска применяется метод Add коллекции PropertyTests.

Общие свойства коллекции PropertyTests

Определения свойств Application, Count и Creator приводились в начале приложения.

Свойства коллекции PropertyTests

Имя	*Тип возвращаемого значения*	*Параметры*	*Описание*
Item	PropertyTest	Index As Long	Возвращает объект PropertyTest из коллекции PropertyTests

Методы коллекции PropertyTests

Имя	*Тип возвращаемого значения*	*Параметры*	*Описание*
Add		Name As String, Condition As MsoCondition, [Value], [SecondValue], [Connector]	Добавляет объект PropertyTest в коллекцию PropertyTests
Remove		Index As Long	Удаляет объект PropertyTest из коллекции PropertyTests

Пример: коллекция PropertyTests

Следующая подпрограмма запрашивает у пользователя учетный номер и находит все книги Excel в указанной папке, у которых значение свойства документа BillingNumber больше или равно ответу пользователя.

```
Sub SearchBillingNumber()

    Dim lCount As Long, lNumber As Long
    Dim oFiles As FoundFiles
    Dim sFiles As String

    ' Запросить нижнюю границу поиска по учетному номеру
    lNumber = Application.InputBox( _
              "С какого учетного номера (начиная с 1)" & _
              "начинать поиск?", _
              "Поиск учетного номера", , , , , , 1)

    ' Если пользователь указал номер больше 1...
    If lNumber > 0 Then
        ' Предположить, но не найдено ни одного файла
        sFiles = "Файлы не найдены"

        ' Воспользоваться функцией ReturnBillingNumbers для получения
        ' списка файлов учета с номером больше, чем значение INumber
        Set oFiles = ReturnBillingNumbers(lNumber)
```

```
        ' Сообщить результат
        If Not oFiles Is Nothing Then
            sFiles = "Найдено файлов:" & vbCrLf
            For lCount = 1 To oFiles.Count
                sFiles = sFiles & vbCrLf & oFiles(lCount)
            Next lCount
        End If

        ' Показать результаты поиска
        MsgBox sFiles, vbInformation, "Billing Files"

    End If
End Sub

Function ReturnBillingNumbers(lNumber As Long) As FoundFiles

    With Application.FileSearch

        ' Сбросить предыдущий поиск, включая значения свойств
        ' FileTypes и Lookin, а также условия свойств.
        ' Пример этой подпрограммы показан в разделе с описанием
        ' коллекции FileTypes
        ResetFileSearch

        ' Добавить первое условие: только книги Excel.
        ' Воспользоваться значением And параметра Connector для
        ' удовлетворения обоих условий
        .PropertyTests.Add _
            Name:="Files of Type", _
            Condition:=msoConditionFileTypeExcelWorkbooks, _
            Connector:=msoConnectorAnd

        ' Добавить второе условие: значение свойства BillingNumber
        ' больше, чем значение lNumber
        .PropertyTests.Add _
            Name:="BillingNumber", _
            Condition:=msoConditionMoreThan, _
            Value:=lNumber

        ' Поиск в папке Billing
        .LookIn = "C:\Billing"

        ' Выполнить поиск и вернуть результаты
        If .Execute > 0 Then
            Set ReturnBillingNumbers = .FoundFiles
        End If

    End With

End Function
```

Объект PropertyTest

Соответствует единственному критерию в коллекции PropertyTests.

Общие свойства объекта PropertyTest

Определения свойств Application, Creator и Parent приводились в начале приложения.

Свойства объекта PropertyTest

Имя	*Тип возвращаемого значения*	*Описание*
`Condition`	`MsoCondition`	Только чтение. Возвращает условие указанного критерия поиска
`Connector`	`MsoConnector`	Только чтение. Возвращает соединитель между двумя похожими значениями условиями. По умолчанию это свойство установлено в значение `msoConnectorAnd`
`Name`	`String`	Только чтение. Возвращает имя объекта `PropertyTest`
`SecondValue`	`Variant`	Только чтение. Возвращает необязательное второе условие для поиска файлов
`Value`	`Variant`	Только чтение. Возвращает значение условия для поиска файлов

Пример: объект PropertyTest

Следующая подпрограмма удаляет объект `PropertyTest`, предназначенный для поиска книг Excel.

```
Sub RemoveExcelPropertyTest()

    Dim lItem As Long
    Dim oProperty As PropertyTest

    ' Инициализировать счетчик
    lItem = 1

    With Application.FileSearch
        ' Перебрать элементы коллекции PropertyTests
        For Each oProperty In .PropertyTests
            ' Удалить условие поиска книг Excel
            If oProperty.Condition = msoConditionFileTypeExcelWorkbooks
            Then
                .PropertyTests.Remove lItem
                Exit For
            Else
                lItem = lItem + 1
            End If
        Next oProperty
    End With
End Sub
```

Коллекция ScopeFolders

В этой коллекции хранится список подпапок для объекта `ScopeFolder`. Данная коллекция используется коллекцией `SearchFolders` объекта `FileSearch` для определения списка папок и подпапок, в которых выполняется поиск. Каждый элемент коллекции `ScopeFolders` является объектом `ScopeFolder` и может использоваться (но не обязательно) в процессе поиска. Каждый объект `ScopeFolder` с подпапками содержит собственный экземпляр коллекции `ScopeFolders` (как подпапки, в которых присутствуют другие подпапки).

Общие свойства коллекции ScopeFolders

Определения свойств Application, Count и Creator приводились в начале приложения.

Свойства коллекции ScopeFolders

Имя	*Тип возвращаемого значения*	*Параметры*	*Описание*
Item	ScopeFolder	Index As Long	Возвращает объект ScopeFolder, соответствующий подпапкам родительского объекта

Объект ScopeFolder

Соответствует одной папке из коллекции ScopeFolders. Каждый объект ScopeFolder содержит собственную коллекцию ScopeFolders, которая соответствует подпапкам папки из объекта ScopeFolder. И объект ScopeFolder, и коллекцию ScopeFolders можно анализировать для определения их применимости в процессе поиска с помощью объекта FileSearch. Любой интересующий во время поиска объект ScopeFolder добавляется в коллекцию SearchFolders с помощью метода AddToSearchFolders объекта ScopeFolder.

Общие свойства объекта ScopeFolder

Определения свойств Application и Creator приводились в начале приложения.

Свойства объекта ScopeFolder

Имя	*Тип возвращаемого значения*	*Описание*
Name	String	Только чтение. Возвращает имя объекта ScopeFolder
Path	String	Только чтение. Возвращает полный путь из объекта ScopeFolder
ScopeFolders	ScopeFolders	Возвращает коллекцию ScopeFolders. Элементы этой коллекции соответствуют подпапкам родительского объекта ScopeFolder

Методы объекта ScopeFolder

Имя	*Тип возвращаемого значения*	*Параметры*	*Описание*
AddToSearch-Folders			Добавляет объект ScopeFolder в коллекцию SearchFolders

Пример: объект ScopeFolder

Следующие подпрограммы выводят список книг Excel из всех папок на жестком диске, имя которых содержит слово “billing”.

```
Sub FindBillingFiles()

    Dim lCount As Long
    Dim oSearchFolders As SearchFolders
    Dim sFiles As String
```

```
    With Application.FileSearch

        ' Сбросить поиск. Дополнительная информация об этой
        ' подпрограмме приводилась в разделе, посвященному коллекции
        ' FileTypes
        ResetFileSearch

        ' Начать поиск книг Excel
        .FileType = msoFileTypeExcelWorkbooks
        ' Искать все папки, имя которых содержит слово "Billing"
        Set oSearchFolders = BillingFolders("RCOR HD (C:)",
↳ "Billing")
        If Not oSearchFolders Is Nothing Then
            .LookIn = oSearchFolders(1)
        End If

        ' Предположить, что файлы не найдены
        sFiles = "Файлы не найдены"

        ' Если файлы найдены, вывести их список
        If .Execute > 0 Then
            sFiles = "Файлы найдены:" & vbCrLf
            For lCount = 1 To .FoundFiles.Count
                sFiles = sFiles & vbCrLf & .FoundFiles(lCount)
            Next lCount
        End If

    End With

    ' Вывести результат работы подпрограммы
    MsgBox sFiles, vbInformation, "Файлы учета"

End Sub
```

Данная функция вызывается основной подпрограммой `FindBillingFiles` и создает коллекцию `SearchFolders` для объекта `FileSearch`.

```
Function oSetSearchFolders(sDrive As String, _
                          sKeyName As String) As SearchFolders

    Dim lCount As Long
    Dim oSearchScope As Searchscope
    Dim oScopeFolder As ScopeFolder

    With Application.FileSearch

        ' Искать только на локальном компьютере (не включая Сетевое
        ' окружение)
        For Each oSearchScope In .SearchScopes
            ' Искать только на локальном жестком диске
            If oSearchScope.Type = msoSearchInMyComputer Then
                ' Перебрать все объекты ScopeFolder из коллекции
                ' ScopeFolders, находящейся в объекте SearchScope
                For Each oScopeFolder In _
                            oSearchScope.ScopeFolder.ScopeFolders
                    If oScopeFolder.Name = sDrive Then
                        ' Эта функция добавляет в коллекцию SearchFolders
                        ' все папки, имя которых содержит слово "Billing"
                        AddFolders oScopeFolder.ScopeFolders, sKeyName
                    End If
                Next oScopeFolder
            End If
```

```
        Next oSearchScope
        ' Если найдена хотя бы одна папка учета, передать список
        ' папок в вызвавшую подпрограмму
        If .SearchFolders.Count > 0 Then
            Set oSetSearchFolders = .SearchFolders
        End If

    End With

End Function
```

Эта подпрограмма вызывается функцией `oSetSearchFolders` и добавляет объект `ScopeFolder` в коллекцию `SearchFolders`.

```
Sub AddFolders(ByVal oScopeFolders As ScopeFolders, ByRef sFolder As _
 String)

    ' Объявить переменную типа ScopeFolder
    Dim oScopeFolder As ScopeFolder

   ' Перебрать все объекты ScopeFolder в коллекции ScopeFolders
   For Each oScopeFolder In oScopeFolders
       ' Не искать в каталогах WINNT и Windows
       If LCase(oScopeFolder.Name) <> "winnt" And _
             LCase(oScopeFolder.Name) <> "windows" Then
           ' Сравнить имя папки из объекта ScopeFolder со значением
           ' переменной sFolder. Воспользоваться функцией LCase
           ' для обеспечения нечувствительности к регистру имени
           If InStr(1, LCase(oScopeFolder.Name), _
                   LCase(sFolder), vbTextCompare) > 0 Then

               ' Добавить объект ScopeFolder в коллекцию
               ' ScopeFolders
               oScopeFolder.AddToSearchFolders

           End If

            ' Разрешить обработку событий
            DoEvents

           ' Если текущий объект ScopeFolder содержит коллекцию
           ' ScopeFolders (подпапки)...
           ' Подавить сообщения об ошибках, возникающие при
           ' обработке папок, которые не воспринимаются как папки
           On Error Resume Next
           If oScopeFolder.ScopeFolders.Count > 0 Then
               ' Вызвать эту подпрограмму рекурсивно для обработки
               ' подпапок
               AddFolders oScopeFolder.ScopeFolders, sFolder
           End If

           On Error GoTo 0

       End If

   Next oScopeFolder

End Sub
```

Коллекция Scripts

Представляет все сценарии в пределах документа, например, в пределах листа Excel, слайда PowerPoint или документа Word. Сценарии представляют собой блоки кода, написанные на ASP, Java, Visual Basic или любом другом языке программирования, работа которого поддерживается в среде HTML. Сценарии запускаются при отображении документов в формате HTML.

Описание создания сценариев HTML выходит за пределы тематики данной книги.

Общие свойства коллекции Scripts

Определения свойств `Application`, `Count`, `Creator` и `Parent` приводились в начале приложения.

Методы коллекции Scripts

Имя	*Тип возвращаемого значения*	*Параметры*	*Описание*
`Add`	`Script`	`[Anchor], [Location], [Language], [Id], [Extended], [ScriptText]`	Добавляет объект `Script` в коллекцию Scripts, которая хранится в пределах объекта `Worksheet` или `Chart`
`Delete`			Удаляет все сценарии из указанного листа
`Item`	`Script`	`Index As Variant`	Возвращает указанный элемент коллекции `Scripts`

Пример: коллекция Scripts

Следующая подпрограмма добавляет простой сценарий в тело листа (в ячейку A1). Лист называется Income Report. При отображении страницы в обозревателе или в режиме предварительного просмота Web-документа в Excel выводится простое окно сообщения, предупреждающее пользователя о том, что диаграммы построены на прошлогодних данных.

```
Sub AddScriptToReport()

   Const sSCRIPT_NAME As String = "DataWarning"

   Dim oScript As Script

   On Error Resume Next
      ThisWorkbook.Worksheets("Income _
Report").Scripts(sSCRIPT_NAME).Delete
   On Error GoTo 0

   ThisWorkbook.Worksheets("Income Report").Scripts.Add _
      Anchor:=ThisWorkbook.Worksheets("Income Report").Range("A1"), _
      Location:=msoScriptLocationInBody, _
      ID:=sSCRIPT_NAME, _
      ScriptText:="MsgBox ""В этом документе используются
↳ прошлогодние данные"""

End Sub
```

Объект Script

Объект Script соответствует одному блоку кода HTML в пределах коллекции Scripts. Для обращения к объему Script можно использовать метод Item коллекции Scripts, например:

```
ThisWorkbook.Worksheets("Income Report").Scripts(1)
```

или

```
ThisWorkbook.Worksheets("Income Report").Scripts.Item(1)
```

Кроме этого, для получения ссылки на объект Script можно воспользоваться идентификатором объекта (ID).

```
ThisWorkbook.Worksheets("Income Report").Scripts("DataWarning")
```

Общие свойства объекта Script

Определения свойств Application, Creator и Parent приводились в начале приложения.

Свойства объекта Script

Имя	*Тип возвращаемого значения*	*Описание*
Extended	String	Устанавливает/возвращает атрибуты, добавленные в тег <SCRIPT>, за исключением атрибутов ID и LANGUAGE
Id	String	Устанавливает/возвращает идентификатор объекта Script. Идентификатор присваивается атрибуту ID в теге <SCRIPT> в коде HTML. Если значение атрибута не указано, свойство возвращает пустую строку
Language	MsoScript-Language	Устанавливает/возвращает язык программирования, на котором написан текущий сценарий
Location	MsoScript-Location	Только чтение. Возвращает положение текущего сценария в указанном документе HTML
ScriptText	String	Устанавливает/возвращает текст, содержащийся в блоке сценария
Shape	Object	Только чтение. Возвращает объект Shape

Методы Script

Имя	*Тип возвращаемого значения*	*Параметры*	*Описание*
Delete			Удаляет указанный сценарий из коллекции Scripts

Пример: объект Script

Следующая подпрограмма удаляет все сценарии со всех листов (листов диаграмм или листов электронных таблиц) в пределах активной книги.

```
Sub RemoveScripts()

    Dim oScript As Script
    Dim sh As Object
```

```
Sub RemoveScripts()

    Dim oScript As Script
    Dim sh As Object

    ' Suppress errors (ex: no active workbook)
    On Error Resume Next

    ' Remove all of the scripts objects in this workbook
    For Each sh In ActiveWorkbook.Sheets
        For Each oScript In sh.Scripts
            oScript.Delete
        Next oScript
    Next sh

End Sub
```

SearchFolders Collection Object

Represents all of the folders used in a File Search (by the `FileSearch` object). `SearchFolders` consist of `ScopeFolder` objects (with the corresponding `ScopeFolders` collection), which are simply folders. Use the `Add` method of the `SearchFolders` object to add `ScopeFolder` objects to its collection.

SearchFolders Collection Common Properties

The `Application`, `Count`, and `Creator` properties are defined at the beginning of this Appendix.

SearchFolders Collection Properties

Name	Returns	Parameters	Description
`Item`	`Scope Folder`	`Index` as `Long`	Returns a `ScopeFolder` object that represents a subfolder of the parent object

SearchFolders Collection Methods

Name	Returns	Parameters	Description
`Add`		`ScopeFolder` as `ScopeFolder`	Adds a search folder to a file search
`Remove`		`Index As Long`	Removes a search folder from a file search

Example: SearchFolders Collection Object

The following routine searches for all Excel workbooks in folders containing the word "billing" on the F drive (on the network) and displays the results on a worksheet:

```
Sub ReportBillingFilesFromNetWorkDrive()

    Dim lCount As Long, lRow As Long
```

```
    Dim oSearchScope As SearchScope
    Dim oScopeFolder As ScopeFolder
    Dim oScopeSubFolder As ScopeFolder

    With Application.FileSearch

        ' Reset FileSearch object. See FileTypes collection for this routine
        ResetFileSearch

        ' Loop through the SearchScopes collection
        ' looking for the My Computer area (scope)
        For Each oSearchScope In .SearchScopes
            If oSearchScope.Type = msoSearchInMyComputer Then
                Set oScopeFolder = oSearchScope.ScopeFolder
                Exit For
            End If
        Next oSearchScope

        ' Now loop through the My computer area (scope)
        ' until we find the mapped "F Billing" drive on the network
        For Each oScopeSubFolder In oScopeFolder.ScopeFolders
            If oScopeSubFolder.Name = "Billing (F:)" Then
                Set oScopeFolder = oScopeSubFolder
                Exit For
            End If
        Next oScopeSubFolder

        ' Now loop through each top-level folder in the F drive adding any
        ' folder that contains the name Billing to the SearchFolders
        ' collection
        Set oScopeSubFolder = Nothing
        For Each oScopeSubFolder In oScopeFolder.ScopeFolders
            If InStr(1, oScopeSubFolder.Name, _
                    "billing", vbTextCompare) > 0 Then
                .SearchFolders.Add oScopeSubFolder
            End If
        Next oScopeSubFolder

        ' Look for Excel workbooks
        .FileType = msoFileTypeExcelWorkbooks

        ' Don't search subfolders. Setting this to True will override the
        ' SearchFolders collection and will search the entire contents of
        ' the F drive
        .SearchSubFolders = False

        If .Execute > 0 Then 'Files were found
            ' Dump the files found on the wksBillingFiles worksheet
            wksBillingFiles.UsedRange.Offset(1, 0).ClearContents

            ' The first row contains the column heading
            lRow = 2
            For lCount = 1 To .FoundFiles.Count
                wksBillingFiles.Cells(lRow, 1).Value = .FoundFiles(lCount)
```

```
                lRow = lRow + 1
            Next lCount
        Else
            MsgBox "No Files found", vbInformation, "Billing Files"
        End If

    End With

End Sub
```

SearchScopes Collection Object

Represents the list of top-level searchable areas when performing a File Search using the `FileSearch` object. Top level areas include My Computer, My Network Places, Outlook (folders), and Custom, if available.

SearchScopes Collection Common Properties

The `Application`, `Count`, and `Creator` properties are defined at the beginning of this Appendix.

SearchScopes Collection Properties

Name	Returns	Parameters	Description
`Item`	`SearchScope`	`Index as Long`	Returns a `SearchScope` object that corresponds to an area in which to perform a file search, such as local drives or Microsoft Outlook folders

Example: SearchScopes Collection Object

The following routine lists all of the `SearchScopes` on the current computer.

```
Sub FindSearchScopes()

    Dim lRow As Long
    Dim oSearchScope As SearchScope

    With Application.FileSearch

        ' Clear the old results
        wksSearchScopes.Range("Info").ClearContents

        ' Set the starting row
        lRow = wksSearchScopes.Range("Info").Cells(1, 1).Row

        ' Loop through the SearchScopes collection
        ' looking for the My Computer area (scope)
        For Each oSearchScope In .SearchScopes
            ' Dump the info found on the wksSearchScopes worksheet
```

```
            wksSearchScopes.Cells(lRow, 1).Value = _
                oSearchScope.ScopeFolder.Name
            wksSearchScopes.Cells(lRow, 2).Value = oSearchScope.Type
            lRow = lRow + 1
        Next oSearchScope

        ' Sort the list by Scope Type
        With wksSearchScopes
            .Range("Info").Sort Key1:=.Cells(1, 2), Order1:=xlAscending
        End With

    End With

End Sub
```

SearchScope Object

An individual top-level area in the `SearchScopes` Collection object that can be searched when using the `FileSearch` object.

SearchScope Common Properties

The `Application` and `Creator` properties are defined at the beginning of this Appendix.

SearchScope Properties

Name	Returns	Description
`ScopeFolder`	`ScopeFolder`	Returns a `ScopeFolder` object
`Type`	`MsoSearchIn`	Read Only. Returns a value that corresponds to the type of `SearchScope` object. The type indicates the area in which the `Execute` method of the `FileSearch` object will search for files

SharedWorkspace Object

The `SharedWorkspace` property returns a `SharedWorkspace` object which allows the developer to add the active document to a Microsoft Windows SharePoint Services document workspace on the server and to manage other objects in the shared workspace.

SharedWorkspace Properties

Name	Returns	Description
`Application`	`Object`	Returns an **`Application`** object that represents the container application this object
`Connected`	`Boolean`	Indicates if the current document is save in and connected to a workspace. Read-only

Общие свойства коллекции SearchScopes

Определения свойств Application, Count и Creator приводились в начале приложения.

Свойства коллекции SearchScopes

Имя	*Тип возвращаемого значения*	*Параметры*	*Описание*
Item	SearchScope	Index As Long	Возвращает объект SearchScope, соответствующий интересующей области поиска, например, локальным дискам или папкам Microsoft Outlook

Пример: коллекция SearchScopes

Следующая подпрограмма выводит список объектов SearchScope в пределах используемого компьютера.

```
Sub FindSearchScopes()

    Dim lRow As Long
    Dim oSearchScope As SearchScope

      With Application.FileSearch

        ' Удалить старые результаты
        wksSearchScopes.Range("Info").ClearContents

        ' Выбрать начальную строку
        lRow = wksSearchScopes.Range("Info").Cells(1, 1).Row

        ' Перебрать коллекцию SearchScopes в поисках области My
        ' Computer
        For Each oSearchScope In .SearchScopes
            ' Вывести полученную информацию на лист wksSearchScopes
            wksSearchScopes.Cells(lRow, 1).Value = _
                oSearchScope.ScopeFolder.Name
            wksSearchScopes.Cells(lRow, 2).Value = oSearchScope.Type
            lRow = lRow + 1
        Next oSearchScope

        ' Отсортировать список по типу области
        With wksSearchScopes
            .Range("Info").Sort Key1:=.Cells(1, 2),
↳ Order1:=xlAscending
        End With

    End With

End Sub
```

Объект SearchScope

Отдельная область верхнего уровня из коллекции SearchScopes, в которой может выполняться поиск с помощью объекта FileSearch.

Общие свойства SearchScope

Определения свойств Application и Creator приводились в начале приложения.

Свойства объекта SearchScope

Имя	*Тип возвращаемого значения*	*Описание*
ScopeFolder	ScopeFolder	Возвращает объект ScopeFolder
Type	MsoSearchIn	Только чтение. Возвращает значение, соответствующее типу объекта SearchScope. Тип определяет область, в которой метод Execute объекта FileSearch будет выполнять поиск файлов

Объект SharedWorkspace

Объект SharedWorkspace позволяет разработчику добавлять активный документ в рабочее пространство сервера Microsoft Windows SharePoint Services, а также поддерживает управление другими объектами в совместном рабочем пространстве.

Свойства объекта SharedWorkspace

Имя	*Тип возвращаемого значения*	*Описание*
Application	Object	Возвращает объект Application, соответствующий приложению, в котором хранится текущий объект
Connected	Boolean	Только чтение. Возвращает информацию о состоянии подключения текущего документа к рабочему пространству
Creator	Long	Возвращает значение типа Long, указывающее на приложение, создавшее объект
Files	SharedWorkspaceFiles	Только чтение. Возвращает коллекцию SharedWorkspaceFiles
Folders	SharedWorkspaceFolders	Только чтение. Возвращает коллекцию SharedWorkspaceFolders
LastRefreshed	Date/Time	Только чтение. Возвращает дату и время последнего вызова метода Refresh
Links	SharedWorkspaceLinks	Только чтение. Возвращает коллекцию SharedWorkspaceLinks
Members	SharedWorkspaceMembers	Только чтение. Возвращает коллекцию SharedWorkspaceMembers
Name	String	Устанавливает/возвращает имя объекта
Parent	Object	Возвращает ссылку на родительский объект текущего объекта
SourceURL	String	Только чтение. Указывает расположение совместно используемого документа
Tasks	SharedWorkspaceTasks	Только чтение. Возвращает коллекцию SharedWorkspaceTasks
URL	String	Только чтение. Возвращает адрес URL общего рабочего пространства

Методы объекта SharedWorkspace

Имя	Тип возвращаемого значения	Параметры	Описание
CreateNew		[URL], [Name]	Создает новый объект SharedWorkspace
Delete			Удаляет объект SharedWorkspace из коллекции
Disconnect			Отключает объект от общего рабочего пространства
Refresh			Обновляет текущую копию документа на основе данных из общего рабочего пространства
Remove-Document			Удаляет текущий документ из общего рабочего пространства

Объект SharedWorkspaceFile

Объект SharedWorkspaceFile соответствует сохраненному файлу из общего рабочего пространства, в качестве которого обычно выступает сервер Sharepoint.

Свойства объекта SharedWorkspaceFile

Имя	Тип возвращаемого значения	Описание
Application	Object	Возвращает объект Application, соответствующий приложению, в котором хранится текущий объект
CreatedBy	String	Только чтение. Возвращает имя автора
CreatedDate	Variant	Только чтение. Возвращает дату создания
Creator	Long	Возвращает значение типа Long, указывающее на приложение, создавшее объект
ModifiedBy	String	Только чтение. Возвращает автора изменений
ModifiedDate	Variant	Только чтение. Возвращает дату модификации
Parent	Object	Возвращает ссылку на родительский объект этого объекта
URL	String	Только чтение. Возвращает адрес URL

Методы объекта SharedWorkspaceFile

Имя	Тип возвращаемого значения	Параметры	Описание
Delete			Удаляет объект SharedWorkspaceFile

Объект SharedWorkspaceFolder

Представляет папку в общем рабочем пространстве.

Свойства объекта SharedWorkspaceFolder

Имя	*Тип возвращаемого значения*	*Описание*
Application	Object	Возвращает объект Application, соответствующий приложению, в котором хранится текущий объект
Creator	Long	Возвращает значение типа Long, указывающее на приложение, создавшее объект
FolderName	String	Только чтение. Возвращает имя папки
Parent	Object	Возвращает ссылку на родительский объект этого объекта

Методы объекта SharedWorkspaceFolder

Имя	*Тип возвращаемого значения*	*Параметры*	*Описание*
Delete			Удаляет текущий объект

Объект SharedWorkspaceLink

Представляет ссылку URL, которая сохраняется в общем рабочем пространстве.

Свойства объекта SharedWorkspaceLink

Имя	*Тип возвращаемого значения*	*Описание*
Application	Object	Возвращает объект Application, соответствующий приложению, в котором хранится текущий объект
CreatedBy	String	Только чтение. Возвращает имя автора
CreatedDate	Variant	Только чтение. Возвращает дату создания
Creator	Long	Возвращает значение типа Long, указывающее на приложение, создавшее объект
ModifiedBy	String	Только чтение. Возвращает автора изменений
ModifiedDate	Variant	Только чтение. Возвращает дату модификации
Description	String	Устанавливает/возвращает описание текущего объекта
Notes	String	Устанавливает/возвращает замечания, связанные с объектом
Parent	Object	Возвращает ссылку на родительский объект этого объекта
URL	String	Только чтение. Возвращает адрес URL

Методы объекта SharedWorkspaceLink

Имя	*Тип возвращаемого значения*	*Параметры*	*Описание*
Delete			Удаляет текущий объект
Save			Сохраняет текущий объект

Объект SharedWorkspaceMember

Представляет пользователя с правами доступа к общему рабочему пространству.

Свойства объекта SharedWorkspaceMember

Имя	*Тип возвращаемого значения*	*Описание*
Application	Object	Возвращает объект Application, соответствующий приложению, в котором хранится текущий объект
Creator	Long	Возвращает значение типа Long, указывающее на приложение, создавшее объект
DomainName	String	Только чтение. Возвращает имя домена
Email	String	Только чтение. Возвращает адрес электронной почты
Name	String	Только чтение. Возвращает имя пользователя
Parent	Object	Возвращает ссылку на родительский объект этого объекта

Методы объекта SharedWorkspaceMember

Имя	*Тип возвращаемого объекта*	*Параметры*	*Описание*
Delete			Удаляет текущий объект

Коллекция SharedWorkspaceMembers

Содержит объекты SharedWorkspaceMember.

Объект SharedWorkspaceTask

Соответствует задаче, сохраненной в общем рабочем пространстве.

Свойства SharedWorkspaceTask

Имя	*Тип возвращаемого значения*	*Описание*
Application	Object	Возвращает объект Application, соответствующий приложению, в котором хранится текущий объект
AssignedTo	String	Устанавливает/возвращает имя пользователя, которому назначена эта задача
CreatedBy	String	Только чтение. Возвращает имя автора
CreatedDate	Variant	Только чтение. Возвращает дату создания
Creator	Long	Возвращает значение типа Long, указывающее на приложение, создавшее объект
Description	String	Устанавливает/возвращает описание текущего объекта
DueDate	Variant	Устанавливает/возвращает конечную дату выполнения задачи
ModifiedBy	String	Только чтение. Возвращает автора изменений
ModifiedDate	Variant	Только чтение. Возвращает дату модификации
Parent	Object	Возвращает ссылку на родительский объект этого объекта

Имя	Тип возвращаемого значения	Описание
Priority	msoShared-WorkspaceTaskPriority	Устанавливает/возвращает приоритет, присвоенный текущей задаче
Status	msoShared-WorkspaceTaskStatus	Устанавливает/возвращает состояние текущей задачи
Title	String	Устанавливает/возвращает название текущей задачи

Методы объекта SharedWorkspaceTask

Имя	Тип возвращаемого значения	Параметры	Описание
Delete			Удаляет текущий объект
Save			Сохраняет текущий объект

Коллекция SharedWorkspaceTasks

Содержит объекты SharedWorkspaceTask.

Объект Signature

Этот объект представляет цифровую подпись документа. Она является электронным вариантом рукописной подписи. Цифровая подпись позволяет другим пользователям документа уникально идентифицировать и проверить источник документа.

При открытии подписанного цифровой подписью документа, содержащего макросы, пользователи, открывающие документ, получают возможность доверять автору или источнику подписи. При установке высокой безопасности использования макросов в пакет Office включаются макросы только из доверенных источников. При этом пользователь может сам выбрать доверенные источники макросов и защитить себя от макровирусов.

Кроме того, цифровая подпись позволяет защитить автора документа от модификации содержимого документа. При подписывании документа к подписи добавляется зашифрованный ключ. При попытке модификации документа другие пользователи получают запрос на предоставление ключа для разблокирования подписи. При этом цифровая подпись с документа снимается.

На данный момент этот объект недоступен в Microsoft Excel, хотя он и предоставляется через объект Document *в Microsoft Word и объект* Presentation *в Microsoft PowerPoint.*

Общие свойства объекта Signature

Определения свойств Application, Creator и Parent приводились в начале этого приложения.

Свойства объекта Signature

Имя	Тип возвращаемого значения	Описание
Attach-Certificate	Boolean	Устанавливает/возвращает необходимость присоединения к документу цифрового сертификата, соответствующего указанному объекту Signature

Имя	*Тип возвращаемого значения*	*Описание*
ExpireDate	Variant	Только чтение. Возвращает дату истечения срока действия цифровой подписи из объекта Signature
IsCertificateExpired	Boolean	Только чтение. Возвращает состояние сертификата, соответствующего объекту Signature (истек ли срок действия сертификата)
IsCertificateRevoked	Boolean	Только чтение. Возвращает состояние сертификата, соответствующего объекту Signature (отозван ли сертификат)
Issuer	String	Только чтение. Возвращает имя организации, выпустившей цифровой сертификат, который соответствует объекту Signature
IsValid	Boolean	Только чтение. Возвращает результат проверки действительности цифрового сертификата, соответствующего объекту Signature. Сертификат может оказаться недействительным по ряду причин, начиная от истечения срока действия и заканчивая модификацией документа, содержащего сертификат
SignDate	Variant	Только чтение. Возвращает дату и время присоединения цифрового сертификата к документу
Signer	String	Только чтение. Возвращает имя лица, присоединившего цифровой сертификат к документу

Методы объекта Signature

Имя	*Тип возвращаемого значения*	*Параметры*	*Описание*
Delete			Удаляет указанную подпись из коллекции SignatureSet

Коллекция SignatureSet

Содержит все объекты Signature в пределах документа.

Общие свойства коллекции SignatureSet

Определения свойств Application, Count, Creator и Parent приводились в начале приложения.

Свойства коллекции SignatureSet

Имя	*Тип возвращаемого значения*	*Параметры*	*Описание*
Item	Signature	iSig As Long	Возвращает объект Signature, соответствующий одной из цифровых подписей, которой подписан текущий документ

Методы коллекции SignatureSet

Имя	*Тип возвращаемого значения*	*Параметры*	*Описание*
Add	Signature		Возвращает объект Signature, соответствующий новой цифровой подписи
Commit			Сохраняет на диске все изменения текущей коллекции SignatureSet. До вызова этого метода изменения коллекции не сохраняются

Пример: коллекция SignatureSet

Следующая подпрограмма добавляет цифровую подпись к активной презентации PowerPoint. После этого подпрограмма выводит информацию, содержащуюся в выбранной подписи. Обратите внимание, что метод Add предоставляет пользователю возможность выбора цифровой подписи из списка.

```
Sub AddSig()

    Dim lIcon As Long
    Dim ppt As Presentation
    Dim oSignature As Signature
    Dim sInfo As String

    ' Проверить наличие активной презентации
    On Error Resume Next
        Set ppt = ActivePresentation
    On Error GoTo 0
    ' Если активная презентация существует...
    If Not ppt Is Nothing Then

        ' Установить пиктограмму, которая отображается в окне
        ' сообщения с информацией
        lIcon = vbInformation

        ' Добавить подпись к презентации. При этом пользователю
        ' придется выбрать подпись. Выбранная подпись возвращается в виде
        ' объекта oSignature
        Set oSignature = ppt.Signatures.Add

        ' Записать изменения на диск. При этом к имени презентации
        ' будет добавлена строка "Signed"
        ppt.Signatures.Commit

        ' Инициализировать строку сообщения
        sInfo = "Информация о подписи:" & vbCrLf & vbCrLf

        ' Добавить подпись к сообщению
        With oSignature
            sInfo = sInfo & "Источник подписи: " & .Issuer & vbCrLf
            sInfo = sInfo & "Владелец: " & .Signer & vbCrLf
            sInfo = sInfo & "Дата подписания: " & .SignDate & vbCrLf
            sInfo = sInfo & "Дата истечения: " & .ExpireDate
        End With
    Else
        ' Активная презентация отсутствует, поэтому выбрать
        ' пиктограмму восклицательного знака и вывести сообщение
        lIcon = vbExclamation
        sInfo = "Активная презентация отсутствует"
    End If
```

```
    ' Вывести сообщение
    MsgBox sInfo, lIcon, "Добавить подпись"

End Sub
```

Объект SmartDocument

Этот объект используется для управления пакетом расширений XML, связанных с активным документом.

Свойства объекта SmartDocument

Имя	*Тип возвращаемого значения*	*Описание*
Application	Object	Возвращает объект Application, соответствующий приложению, в котором хранится текущий объект
Creator	Long	Возвращает значение типа Long, указывающее на приложение, создавшее объект
SolutionID	String	Устанавливает/возвращает идентификатор решения
SolutionURL	String	Устанавливает/возвращает адрес URL-решения

Методы SmartDocument

Имя	*Тип возвращаемого значения*	*Параметры*	*Описание*
PickSolution		[ConsiderAllSchemas]	Выбирает решение
RefreshPane			Обновляет область задач

Объект Sync

Объект Sync применяется для управления синхронизацией локальной и серверной копий совместно используемого документа, хранящегося в рабочей области сервера Windows SharePoint Services. Свойство Status возвращает важную информацию о текущем состоянии процесса синхронизации. Для обновления состояния синхронизации необходимо воспользоваться методом GetUpdate. Свойства LastSyncTime, ErrorType и WorkspaceLastChangedBy применяются для получения дополнительной информации.

Свойства Sync

Имя	*Тип возвращаемого значения*	*Описание*
Application	Object	Возвращает объект Application, соответствующий приложению, в котором хранится текущий объект
Creator	Long	Возвращает значение типа Long, указывающее на приложение, создавшее объект
ErrorType	MsoSyncErrorType	Только чтение. Возвращает тип последней ошибки синхронизации
LastSyncTime	Variant	Только чтение. Возвращает время последней синхронизации
Parent	Object	Возвращает ссылку на родительский объект этого объекта

Имя	*Тип возвращаемого значения*	*Описание*
`Status`	`MsoSyncStatusType`	Только чтение. Возвращает текущее состояние синхронизации
`WorkspaceLastChangedBy`	`String`	Только чтение. Возвращает автора последней модификации

Методы объекта Sync

Имя	*Тип возвращаемого значения*	*Параметры*	*Описание*
`GetUpdate`			Получает обновления с сервера
`OpenVersion`		`[SyncVersionType]`	Открывает указанную версию документа
`PutUpdate`			Размещает обновление на сервере
`ResolveConflict`		`[SyncConflictResolution]`	Запускает процедуру разрешения конфликта при обновлении документа
`Unsuspend`			Отключает приостановленное состояние документа

Объект UserPermission

Представляет набор разрешений на доступ для одного пользователя.

Свойства объекта UserPermission

Имя	*Тип возвращаемого значения*	*Описание*
`ExpirationDate`	`Variant`	Устанавливает/возвращает дату истечения разрешения на доступ
`Permission`	`Long`	Устанавливает/возвращает разрешение на доступ
`UserId`	`String`	Только чтение. Возвращает идентификатор пользователя

Методы объекта UserPermission

Имя	*Тип возвращаемого значения*	*Параметры*	*Описание*
`Remove`			Удаляет текущий объект `UserPermission` из коллекции `Permissions`

Коллекция WebPageFonts

Представляет набор объектов `WebPageFont`, позволяющих устанавливать стили и размеры моноширинных и пропорциональных шрифтов при сохранении документа в виде Web-страницы. Дополнительная информация приводится в разделе, посвященном объекту `WebPageFont`.

К коллекции можно обращаться через свойство `Fonts` свойства `DefaultWebOptions` объекта `Application`, например:

```
Set oWebPageFonts = Application.DefaultWebOptions.Fonts
```

Обратите внимание, что на момент написания этой книги свойство Count коллекции WebPageFont всегда возвращает значение 0, даже если в коллекции присутствует 12 объектов WebPageFont.

Общие свойства коллекции WebPageFonts

Определения свойств Application, Count и Creator приводились в начале приложения.

Свойства коллекции WebPageFonts

Имя	*Тип возвращаемого значения*	*Параметры*	*Описание*
Item	WebPageFont	Index As MsoCharacterSet	Возвращает объект WebPageFont из коллекции WebPageFonts, соответствующий определенному набору символов

Объект WebPageFont

Этот объект описывает моноширинные и пропорциональные шрифты, используемые при сохранении документа в виде Web-страницы. Microsoft Excel и Microsoft Word применяют данные параметры при открытии Web-страниц в приложении, но эти параметры используются только при невозможности применения собственных параметров шрифтов Web-страницы или в коде HTML отсутствует информация о необходимых шрифтах.

Обратите внимание, что свойства FixedWidthFont и ProportionalFont принимают любое допустимое значение типа String, а свойства FixedWidthFontSize и ProportionalFontSize принимают любое допустимое значение типа Single. Например, следующий код не приведет к появлению сообщения об ошибке, хотя присваиваемые значения и не описывают допустимые имя и размер шрифта.

```
Application.DefaultWebOptions.Fonts
↳(msoCharacterSetEnglishWesternEuropean
OtherLatinScript).ProportionalFont = "XXXXXXX"
```

```
Application.DefaultWebOptions.Fonts
↳(msoCharacterSetEnglishWesternEuropean
OtherLatinScript).ProportionalFontSize = 1200
```

При попытке использования этих значений в приложении будет выдано сообщение об ошибке.

Общие свойства объекта WebPageFont

Определения свойств Application и Creator приводились в начале приложения.

Свойства объекта WebPageFont

Имя	*Тип возвращаемого значения*	*Описание*
FixedWidth-Font	String	Устанавливает/возвращает имя моноширинного шрифта
FixedWidth-FontSize	Single	Устанавливает/возвращает размер моноширинного шрифта (в пунктах)

Имя	*Тип возвращаемого значения*	*Описание*
`ProportionalFont`	`String`	Устанавливает/возвращает имя пропорционального шрифта
`ProportionalFontSize`	`Single`	Устанавливает/возвращает размер пропорционального шрифта (в пунктах)

Пример: объект WebPageFont

Следующая подпрограмма перебирает коллекцию `WebPageFonts` и для каждого объекта `WebPageFont` указывает имя и размер пропорционального и моноширинного шрифта. Информация об имени и размере выбирается из диапазона `FontInfo` на листе `wksWebPageFonts` из текущей книги.

```
Sub SetWebPageFonts()

    Const sRANGE_FONT_INFO As String = "FontInfo"

    Dim lCalc As Long
    Dim lCount As Long, lRow As Long
    Dim oWebFont As WebPageFont
    ' Отключить обновление экрана и пересчет листа
    With Application
        ' Сохранить режим пересчета листа
        lCalc = .Calculation
        ' Отключить пересчет листа
        .Calculation = xlCalculationManual
        ' Отключить обновление экрана
        .ScreenUpdating = False
    End With

    ' Установить счетчик строк на основе первой ячейки диапазона
    ' ввода
    lRow = wksWebPageFonts.Range("InputRange").Cells(1, 1).Row

    ' Удалить старую информацию
    wksWebPageFonts.Range("InputRange").ClearContents

    ' Существует 12 наборов символов для шрифтов. Приходится жестко
    ' задавать ограничение в 12 наборов, так как свойство
    ' Application.DefaultWebOptions.Fonts.Count всегда возвращает
    ' значение 0
    For lCount = 1 To 12
        ' Получить ссылку на объект WebPageFont
        Set oWebFont = Application.DefaultWebOptions.Fonts(lCount)

        With oWebFont
            ' Воспользоваться параметрами с листа wksWebPageFonts и
            ' установить свойства объекта WebPageFont
            .ProportionalFont = _
                wksWebPageFonts.Range(sRANGE_FONT_INFO).Cells(1,
➥ 1).Value
            .ProportionalFontSize = _
                wksWebPageFonts.Range(sRANGE_FONT_INFO).Cells(1,
➥ 2).Value
            .FixedWidthFont = _
                wksWebPageFonts.Range(sRANGE_FONT_INFO).Cells(1, 3).Value
            .FixedWidthFontSize = _
                wksWebPageFonts.Range(sRANGE_FONT_INFO).Cells(1,
```

```
↳ 4).Value

            ' Вывести новые параметры WebPageFont в таблицу на листе
            ' wksWebPageFonts
            wksWebPageFonts.Cells(lRow, 2) = lCount
            wksWebPageFonts.Cells(lRow, 3) = .ProportionalFont
            wksWebPageFonts.Cells(lRow, 4) = .ProportionalFontSize
            wksWebPageFonts.Cells(lRow, 5) = .FixedWidthFont
            wksWebPageFonts.Cells(lRow, 6) = .FixedWidthFontSize

            ' Переместиться на одну строку вниз
            lRow = lRow + 1
        End With
    Next lCount

    ' Восстановить режим пересчета листа
    Application.Calculation = lCalc

End Sub
```

Предметный указатель

A

API, 366

I

IDTExtensibility2, интерфейс, 299
Internet
  источник данных, 627
  передача данных, 636
  публикация результатов, 633
  хранение книг, 626

M

Microsoft SQL Server, 270

X

XML, 655
  импорт данных, 658
XMLSS, 658
XSD, 656

А

Автофильтр, 519
Агрегация, 141
Аргументы, 133
  необязательные, 134
  передача по значению, 133
  передача по ссылке, 134

В

Вирус, 361
Возвращаемые значения, 76
Всплывающие меню, 563

Д

Диаграммы
  встроенные, 534
  добавление из кода VBA, 533
  использование массивов, 542
  листы, 532
  метки, 544
  определение диапазонов, 543
  определение рядов через массив, 540
  редактирование рядов, 536
Диапазон
  сокращенные ссылки, 451
  ссылка, 456
Документ Word, 356; 357

И

Имена
  диапазонов, 478
  именованные диапазоны, 482
  поиск, 483
  сокрытие, 482
  специальные, 479
  хранение значений, 480
  хранение массивов, 481
Инкапсуляция, 141
Интерфейс, 129; 299
  определение, 131
  реализация, 131; 300

К

Квалификаторы, 140
Класс, 129; 372
  Actions, 610
  Recognizer, 607
Ключевое слово
  static, 108
Книга
  активизация, 434
  перезапись существующей, 437
  получение имени файла, 435
  создание, 433
  сохранение, 434; 439
Коллекция, 64; 155
  Addins, 666; 961
  AllowEditRanges, 668
  AnswerWizardFiles, 989
  Areas, 691
  Axes, 697
  BalloonCheckBoxes, 995

BalloonLabels, 996
Borders, 702
CalculatedFields, 506; 703; 818
CalculatedItems, 512; 704; 823
CalculatedMembers, 704
ChartGroups, 727
ChartObjects, 730
Charts, 711
CodePanes, 967
COMAddins, 998
CommandBarControls, 1014
CommandBars, 1000
Comments, 737
CubeFields, 743
CustomViews, 747
DataLabels, 748
DiagramNodes, 758
Dialogs, 760
DocumentLibraryVersions, 1024
DocumentProperties, 1024
ErrorCheckingOptions, 766
Errors, 764
FileDialogFilters, 559; 1031
FileDialogSelectedItems, 559; 1033
FileTypes, 552; 1038
Filters, 770
FormatConditions, 773
GroupShapes, 779
HPageBreaks, 780
HTMLProjectItems, 1042
Hyperlinks, 781
LegendEntries, 788
LinkedWindows, 971
Names, 796
ODBCErrors, 798
ODSOColumns, 1047
ODSOFilters, 1048
OLEDBErrors, 799
OLEObjects, 801
Panes, 809
Parameters, 810
Phonetics, 812
PivotCaches, 502; 814
PivotFields, 503; 818
PivotFormulas, 823
PivotItems, 508; 823
PivotTables, 503; 827
Points, 837
Properties, 971
PropertyTests, 1050
PublishObjects, 841
QueryTables, 843
RecentFiles, 860
References, 973
Scenarios, 863
ScopeFolders, 1053
Scripts, 1057
SearchFolders, 555; 1059
SearchScopes, 553; 1060
SeriesCollection, 864
ShapeNodes, 875
ShapeRange, 877
Shapes, 870
SharedWorkspaceMembers, 1065
SharedWorkspaceTasks, 1066
Sheets, 439; 880
SignatureSet, 1067
SmartTagActions, 883
SmartTagOptions, 884
SmartTagRecongnizers, 884
SmartTags, 882
SpellingOptions, 886
Styles, 887
Trendlines, 895
UsedObjects, 898
UserAccess, 898
UserAccessList, 899
VBComponents, 977
VPageBreaks, 902
Watches, 904
WebPageFonts, 1070
Windows, 906; 984
Workbooks, 433; 911
Worksheets, 925
в модуле класса, 156
Команда
Add Watch, 196
Bookmark, 202
Call Stack, 204
Complete Word, 202
Definition, 203
Edit Watch, 196
List Constants, 202
List Properties/Methods, 202
ObjectBrowser, 203
Parameter Info, 202
Quick Info, 200
Quick Watch, 196

Run to cursor, 194
Set Next Statement, 195
Show Next Statement, 195
Step Into, 193
Step Out, 194
Step Over, 194
Командные панели, 566
Константы, 76; 84

М

Массив, 95
динамический, 98
многомерный, 97
Меню, 326
Метод, 67
Activate, 449
Application.Evaluate, 405
Debug.Assert, 169
Debug.Print, 168
Evaluate, 118
Execute, 559
InputBox, 120
Intersect, 469
OnKey, 124
OnTime, 123
Recognize, 623
Select, 449
SpecialCells, 460
Union, 469
определение, 132
Модуль класса, 153; 374

Н

Надстройки
закрытие, 289
модификация кода, 289
преобразование книги, 287
событие установки, 291
сохранение, 290
удаление, 292
установки, 290
Надстройки Automation, 293
использование, 298
регистрация, 295
создание, 294
Надстройки COM, 308
конструктор, 311
подключение к Excel, 312
регистрация, 309

О

Область видимости, 81
Обработка ошибок, 99
Объект, 63
Addin, 666; 961
Adjustments, 667
AllowEditRange, 668
AnswerWizard, 988
Application, 115; 670
Assistant, 990
AutoCorrect, 693
AutoFilter, 694
AutoRecover, 696
Axis, 697
AxisTitle, 700
Balloon, 994
BalloonCheckBox, 996
BalloonLabel, 996
Border, 702
CalculatedMember, 704
CalloutFormat, 706
CellFormat, 707
Characters, 710
Chart, 711
ChartArea, 723
ChartColorFormat, 725
ChartFillFormat, 725
ChartGroup, 727
ChartObject, 730
ChartTitle, 735
CodeModule, 324; 963
CodePane, 324; 967
ColorFormat, 736
COMAddin, 999
Command, 258
CommandBar, 1002
CommandBarButton, 1005
CommandBarComboBox, 1009
CommandBarControl, 1015
CommandBarPopup, 1019
Comment, 737
Connection, 244
ConnectorFormat, 738
ControlFormat, 740
Corners, 742

CubeField, 743
CustomView, 747
DataLabel, 748
DataTable, 752
Debug, 168
DefaultWebOptions, 753
Designer, 324
Diagram, 755
DiagramNode, 758
DiagramNodeChildren, 760
Dialog, 760
DisplayUnitLabel, 761
DocumentLibraryVersion, 1023
DocumentProperty, 1026
DownBars, 763
DropLines, 763
Err, 186
Error, 764
ErrorBars, 765
Events, 970
FileDialog, 557; 1028
FileDialogFilter, 1032
FileSearch, 548; 1036
FillFormat, 768
Filter, 770
Floor, 771
Font, 772
FormatCondition, 773
FoundFiles, 1039
FreeformBuilder, 775
Graphic, 776
Gridlines, 778
HiLoLines, 779
HPageBreak, 780
HTMLProject, 1039
HTMLProjectItem, 1042
Hyperlink, 781
Interior, 783
IRtdServer, 784
IRTDUpdateEvent, 785
LanguageSettings, 1043
LeaderLines, 786
Legend, 786
LegendEntry, 788
LegendKey, 789
LineFormat, 791
LinkFormat, 792
ListColumn, 793
ListDataFormat, 793
ListObject, 794
ListRow, 795
Mailer, 795
MsoEnvelope, 1044
Name, 796
NewFile, 1046
ODBCError, 798
ODSOColumn, 1047
ODSOFilter, 1048
OfficeDataSourceObject, 1049
OLEDBError, 799
OLEFormat, 800
OLEObject, 801
Outline, 805
PageSetup, 806
Pane, 809
Parameter, 810
Permission, 1049
Phonetic, 812
PictureFormat, 813
PivotCache, 814
PivotCell, 817
PivotField, 818
PivotFormula, 823
PivotItem, 823
PivotItemList, 825
PivotLayout, 826
PivotTable, 827
PlotArea, 835
Point, 837
Property, 971
Protection, 839
PublishObject, 841
QueryTable, 843
Range, 847
RecentFile, 860
Recordset, 253
Reference, 973
ReferencesEvents, 976
RoutingSlip, 861
RTD, 862
Scenario, 863
ScopeFolder, 1054
Script, 1058
SearchScope, 1061
Series, 864
SeriesLines, 869
ShadowFormat, 869
Shape, 870
ShapeNode, 875

SharedWorkspace, 1062
SharedWorkspaceFile, 1063
SharedWorkspaceFolder, 1063
SharedWorkspaceLink, 1064
SharedWorkspaceMember, 1065
SharedWorkspaceTask, 1065
Signature, 1066
SmartDocument, 1069
SmartTag, 882
SmartTagAction, 883
SmartTagRecongnizer, 884
SoundNote, 885
Speech, 885
Style, 887
Sync, 1069
Tab, 889
TextEffectFormat, 890
TextFrame, 891
ThreeDFormat, 892
TickLabels, 894
TreeviewControl, 895
Trendline, 895
UpBars, 897
UserPermission, 1070
Validation, 900
VBComponent, 323; 977
VBE, 322; 980
VBProject, 322
VPageBreak, 902
Walls, 903
Watch, 904
WebOptions, 905
WebPageFont, 1071
Window, 444; 906; 984
Workbook, 911
Worksheet, 925
WorksheetFunction, 934
XmlDataBinding, 956
XmlMap, 957
создание, 154
Объектные переменные, 85
Объявления в стиле С, 367
Окно
Call Stack, 113
Control Toolbox, 107
Immediate, 71; 200
Locals, 112; 199
Object Browser, 70; 113
Project Explorer, 51; 106
Properties, 51; 109
Watch, 112
Окно UserForm
немодальные окна, 221
отображение, 207
создание, 209
Оператор
Call, 79
CREATE TABLE, 242
DROP TABLE, 243
If, 87
блочный, 88
INSERT, 240
New, 108
On Error GoTo, 183
On Error GoTo 0, 186
On Error Resume Next, 101; 184
Option Explicit, 79
SELECT, 239
Select Case, 89
Set, 108
UPDATE, 241
With... End With, 86
Отключенные наборы записей, 277
Отладка, 170
вывод сообщений об ошибках, 181
журнал событий, 188
контрольные значения, 196
обработчики ошибок, 183
перехват, 176
последовательность выполнения, 173
пошаговое выполнение кода, 192
создание обвязки, 186
точки останова, 195

П

Панели инструментов, 223; 563
Панели меню, 563
Передача параметров
по имени, 74
по позиции, 74
Позднее связывание, 352
Поле, 65
определение, 135
Процедура, 50

Р

Раннее связывание, 349; 354
Раскрывающийся список, 520

Расширенный фильтр, 527
Региональные параметры
  идентификация, 399
  изменение, 398
  обработка, 398
  функции преобразования, 399
Рекурсия
  методы, 134
  отказ от рекурсии, 135
Рефакторинг, 106

С

Сводная диаграмма, 512
Сводная таблица
  отчет, 500
Свойства
  определение, 136
  только запись, 137
  только чтение, 137
Свойство, 66
  ActiveCell, 116
  ActiveChart, 116
  ActivePrinter, 116
  ActiveSheet, 117
  ActiveWindow, 117
  ActiveWorkbook, 117
  Caller, 126
  Cells, 453; 454
  Columns, 466
  CurrentRegion, 462
  End, 464; 465
  FoundFiles, 550
  MultiSelect, 560
  Name, 479
  Offset, 457; 458
  OnAction, 575
  PropertyTests, 551
  Range, 451
  Rows, 466
  ScopeFolders, 554
  ScreenUpdating, 118
  Selection, 117
  SendKeys, 122
  StatusBar, 122
  Visible, 511
Смарт-тег
  добавление, 622
  проблемы, 623
  регистрация, 618
  структура, 605
  удаление, 622
  уникальный идентификатор, 605
  управление из VBA, 621
Событие, 68
  BeforeDoubleClick, 147
  Worksheet Calculate, 145
События
  включение, 145
  диаграмм, 146; 160
  книги, 149
  листа, 144
  обработка, 140
  определение, 137
  определение в классах, 138
  перехват, 158
  создание, 139
Сокрытие кода, 286
Списки
  изменение размера, 492
  преобразование в диапазон, 493
  публикация, 493
  создание, 489
  создание окна UserForm, 491
  сортировка, 490
  сумма элементов, 492
  фильтрация, 490
Ссылки, 163; 348
Структура, 371

Т

Тип переменной, 82
  определение, 84

Ф

Функция
  CBool, 402
  CByte, 402
  CCur, 402
  CDate, 402
  CDbl, 402
  CDec, 402
  CInt, 402
  CLng, 402
  CSng, 402
  CStr, 402
  DateValue, 402
  DateValueUS, 405
  Format, 402
  FormatCurrency, 403

FormatDateTime, 403
FormatNumber, 403
FormatPercent, 403
InputBox, 120
IsDate, 401
IsDateUS, 405
IsNumeric, 401
NumberToString, 403
Str, 403
Val, 404

Х

Хранимые процедуры, 272

Ц

Цикл, 90
Do... Loop, 91
For Each... Next, 95
For... Next, 93
While... Wend, 91

Э

Элемент управления
CheckBox, 227
OptionButton, 227
ScrollBar, 225
SpinButton, 226
Элементы управления ActiveX, 224

Научно-популярное издание

Пол Киммел, Джон Грин, Стивен Буллен, Роб Боуви, Роберт Розенберг и др.

Excel 2003 и VBA
Справочник программиста

Литературный редактор	*Т.Г. Сковородникова*
Верстка	*О.В. Мишутина*
Художественный редактор	*В.Г. Павлютин*
Корректоры	*Л.А. Гордиенко,*
	Т.А. Корзун,
	А.В. Луценко,
	О.В. Мишутина

Издательский дом "Вильямс".
101509, Москва, ул. Лесная, д. 43, стр. 1.

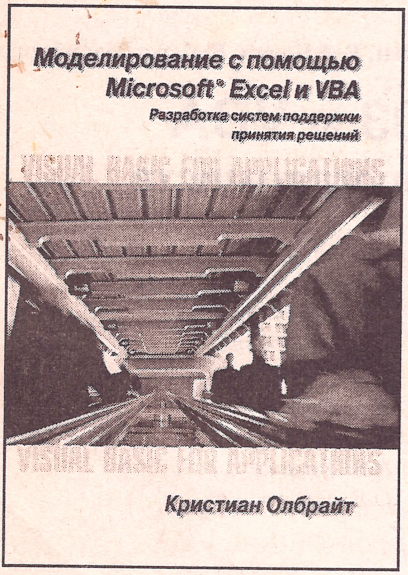
Подписано в печать 23.11.2005. Формат 70×100/16.
Гарнитура Times. Печать офсетная.
Усл. печ. л. 87,72. Уч.-изд. л. 56,6.
Тираж 3000 экз. Заказ № 6844.

Отпечатано с диапозитивов
в ФГУП "Печатный двор" им. А. М. Горького
Федерального агентства по печати и массовым коммуникациям.
197110, Санкт-Петербург, Чкаловский пр., 15.

МОДЕЛИРОВАНИЕ С ПОМОЩЬЮ MICROSOFT EXCEL И VBA

РАЗРАБОТКА СИСТЕМ ПОДДЕРЖКИ ПРИНЯТИЯ РЕШЕНИЙ

Кристиан Олбрайт



www.williamspublishing.com

В этой книге рассказывается о том, как эффективно применять методы моделирования в приложениях электронных таблиц для решения повседневных экономических задач. Книга состоит из двух частей. В первой из них изложены основные концепции и принципы программирования на языке VBA (Visual Basic for Applications), во второй части рассматриваются конкретные приложения и детально анализируется их исходный код. В конце каждой главы приводится ряд упражнений по программированию, которые призваны закрепить или расширить полученные знания и лучше усвоить изученные в главе концепции. Прилагаемый к книге компакт-диск содержит все рабочие книги Excel и другие файлы, о которых речь идет в примерах (в том числе файлы, используемые в упражнениях). Все рабочие книги открываются в Excel 97 или более поздней версии программы. Настоящая книга будет полезна как студентам, так и профессионалам, которые хотят создавать системы поддержки принятия решений с помощью электронных таблиц на основе Microsoft Excel.

ISBN 5-8459-0858-2 **в продаже**

Профессиональное программирование на VBA в Excel 2003

Дж. Уокенбах

ISBN 5-8459-0771-3
в продаже

По Excel написано немало книг. Но книга, которую вы держите в руках, является особенной — в ней разработка приложений электронных таблиц рассматривается в широком контексте. VBA — это всего лишь один из компонентов среды разработки пользовательских приложений, хотя и довольно существенный. Данная книга поможет вам разобраться в тонкостях разработки приложений с помощью VBA. В ней описаны многочисленные средства языка VBA, его возможности и среда использования.

Вначале вашему вниманию будет предложен обзор возможностей программы, далее вы перейдете к определению концепций VBA-программирования, а затем познакомитесь с самим языком. Если вы начинающий программист на VBA, то в данном издании вы найдете всю необходимую информацию, которая потребуется для дальнейшей работы. Если вы уже обладаете завидным опытом работы с VBA, то эта книга обогатит и приумножит ваши знания, пополнив их новыми методиками и примерами из реальной жизни.

http://www.dialektika.com

ПРОГРАММИРОВАНИЕ НА VBA
САМОУЧИТЕЛЬ

Слепцова Л.Д.

www.dialektika.com

Книга предназначена для индивидуального обучения, написана простым, доступным языком и рассчитана на пользователей с любым уровнем компьютерных знаний. Единственное, что действительно необходимо читателю — это наличие под рукой компьютера с установленными программами Microsoft Office (хотя бы одной!) и искреннее стремление научиться программировать на языке VBA. Книга вполне может стать настольным справочником, к которому можно будет обратиться в любой момент, столкнувшись со сложной или непонятной ситуацией. Благодаря лаконичному и, одновременно, исчерпывающему описанию существующих в языке VBA средств и методов работы с ними, вы найдете здесь ответы на любые вопросы, которые могут возникнуть в практике программирования офисных приложений.
В этой книге рассмотрены также темы без проработки которых (хотя формально они и не относятся к самому языку) освоение программирования на VBA и создание реальных программ в среде Office будет просто невозможно.
Книга содержит множество практических примеров.

ISBN 5-8459-0608-3

в продаже

Programmer to Programmer™

Excel 2003 VBA

Programmer's Reference

If you're an Excel user seeking more control over your spreadsheets, or if you are developing Excel applications for others, this comprehensive volume provides what you need to maximize VBA flexibility in the Excel environment. Fully revised and updated, it focuses exclusively on VBA for Excel and provides appropriate information for beginners, Excel programmers, and advanced developers alike.

Packed with examples, code, and screenshots where applicable, this clearly organized reference meets you at the level of your experience and takes you to the next level. From a complete Excel VBA primer to internationalization, advanced debugging techniques, and programming the VB editor, this is your one-stop reference for Excel VBA programming.

What you will learn from this book

- Object-oriented theory and VBA for Excel
- Ways to set up applications and convert them to add-ins
- How to use VB6 and VB.NET with Excel
- Programming to the Windows® API
- Advanced error handling and debugging methods
- Essentials of SQL® as it applies to accessing data with ADO
- Interaction with other Office applications and databases
- How to access extensive reference material in downloadable, searchable format

Who this book is for

This book is for Excel users and programmers from beginning to advanced. You need a reasonable working knowledge of Excel and a full installation of the software.

Wrox Programmer's References are designed to give the experienced developer straight facts on a new technology, without hype or unnecessary explanations. They deliver hard information with plenty of practical examples to help you apply new tools to your development projects today.

p2p.wrox.com
The programmer's resource center

www.wrox.com

7 85555 87949 2

Recommended Computer Book Categories	Programming
	VBA

$39.99 USA
$57.99 CANADA
£26.99 UK

Wrox™
An Imprint of
WILEY

ISBN: 0-7645-5660-6

53999

9 780764 556609